НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Автоматизованих систем обробки інформації і управління
(повна назва кафедри)

«На правах рукопису»
УДК 004.42

До захисту допущено:
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютеризованих систем»

зі спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Екстракція структурованої інформації з множини веб-сторінок»

Виконав (-ла):
студент (-ка) VI курсу, групи ІП-92мп
Смілянець Федір Андрійович _____

Науковий керівник:
Доцент кафедри АСОІУ, к.т.н
Мажара Ольга Олександрівна _____

Рецензент:
Доцент кафедри АПЕПС ТЕФ, к.т.н
Шаповалова Світлана Ігорівна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Автоматизованих систем обробки інформації і управління

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма - «Інженерія програмного забезпечення комп'ютеризованих систем»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Смілянцю Федору Андрійовичу

1. Тема дисертації «Екстракція структурованої інформації з множини веб-сторінок», науковий керівник дисертації Мажара Ольга Олександрівна, к.т.н. затверджені наказом по університету від «_26_»__жовтня_ 2020 р. №3132-с_

2. Термін подання студентом дисертації_____

3. Об'єкт дослідження процес екстракції текстових даних з подальшою_____
обробкою методами машинного навчання_____

4. Вхідні дані множина веб-сторінок новинних ресурсів_____

5. Перелік завдань, які потрібно розробити провести аналіз існуючих рішень у сфері екстракції структурованої інформації з множини веб-сторінок, розробити інструмент екстракції, виконати порівняння інструменту екстракції з існуючими аналогами у вигляді постачання даних для систем машинного навчання_____

6. Орієнтовний перелік графічного (ілюстративного) матеріалу іллюстрації роботи розробленого програмного забезпечення_____

7. Орієнтовний перелік публікацій публікація тез доповіді в матеріалах наукової конференції

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання 06.11

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	06.10	
2	Аналіз існуючих методів	11.10	
3	Вибір інструментарію розробки	18.10	
4	Розробка алгоритму видобуваннння даних	27.10	
5	Розробка імплементації алгоритму	02.11	
6	Розробка споживачів даних	11.10	
7	Тестування систем	15.10	
8	Розробка документації	24.10	
9	Подання роботи на попередній захист	27.11	
10	Подання роботи на основний захист	17.11	

Студент Федір СМІЛЯНЕЦЬ

Науковий керівник Ольга МАЖАРА

РЕФЕРАТ

Актуальність теми дослідження. Сучасний широкий інтернет є істотним джерелом даних для використання у наукових та бізнес-дослідженнях. Можливість видобувати актуальні дані часто є ключовою для досягнення необхідних цілей, але сучасні якісні рішення з застосуванням технологій машинного зору та інших можуть бути дорогими до придбання або розробки, тому прості та дешеві як з точки зору розробки та підтримки, так і з точки зору експлуатації рішення є необхідними.

Метою дослідження є створення програмного інструментарію екстракції структурованих даних з веб-сторінок новинних ресурсів для подальшої класифікації за достовірністю. Для досягнення поставленої мети було окреслено та виконано наступні завдання:

- провести огляд існуючих підходів та програмних аналогів у областях екстракції даних з веб-ресурсів та оцінки якості новин;
- позробити та реалізувати алгоритми екстракції, підготовки та класифікації даних;
- порівняти результати, отримані розробленим алгоритмом та результатами тренування алгоритмів машинного навчання на даних, видобутих ним з існуючим аналогом та результатами тренування на даних аналогу.

Об’єктом дослідження є процес екстракції текстових даних з подальшою обробкою методами машинного навчання.

Предметом дослідження є методи та засоби екстракції та аналізу структурованих текстових даних

Наукова новизна одержаних результатів. Було створено простий жадібний алгоритм у якому суміщено процеси пошуку посилань та видобування інформації, доведено доцільність використання простих алгоритмів для збору даних з ресурсів у мережі Інтернет з ціллю використання у тренуванні алгоритмів машинного навчання.

Було доведено що як класичні алгоритми навчання здатні досягати результатів, співставним з такими у нейронних мереж, таких як мережі ДКЧП, та показано що такі моделі здатні працювати на двомовному датасеті.

Публікації. Матеріали роботи було опубліковано у п’ятій Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020) «Класифікація новин за достовірністю на основі методів машинного навчання»

ВЕБ-СКРАПІНГ, ЕКСТРАКЦІЯ ДАНИХ З ВЕБ-СТОРІНОК, КРАУЛІНГ, ПОШУК ПОСИЛАНЬ, МАШИННЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ НОВИН, ДОВГА КОРОТКОЧАСНА ПАМ’ЯТЬ, НЕЙРОННІ МЕРЕЖІ

ABSTRACT

Relevance of the research topic. Modern wide internet is a considerable source of data to be used in scientific and business applications. An ability to extract up to date data is frequently crutial for reaching necessary goals, though, modern quality solutions to this problem, which are using computer vision and other technologies, may be finantially demanding to acquire or develop, thus simple and cheap to develop, maintain and use solutions are necessary.

The purpose of the study is to create a software instrument aimed at extraction of structured data from news websites for usage in news trustworthiness classification. Following tasks were outlined and implemented to achieve the aforementioned goal:

- Outline existing approaches and analogues in areas of data extraction and news classification;
- Design and develop extraction, preparation and classification algorhitms;
- Compare the results achieved with developed extraction algorhitm and with existing software solution, including comparing machine learning accuracies on both of the extractors.

The object of the study is the process of text data extraction with subsequent machine learning analysis.

The subjects of the study are methods and tools of extraction and analysis of text data.

Scientific novelty of the obtained results. A simple greedy algorithm was created, combining the process of link discovery and data extraction. Expediency of usage of simple web data extraction algorithms for composing machine learning datasets was proven.

It was also proven that classical machine learning algorithms can achieve results similar to neural networks such as LSTM. Capabilities of machine learning systems to function efficiently in a bilingual context were also shown.

Publications. Materials, related to this study, were published in the All-Ukrainian Scientific and Practical Conference of Young Scientists and Students

“Information Systems and Management Technologies” (ISTU-2019) “News trustworthiness classification with machine learning”

WEB SCRAPING, WEB PAGE DATA EXTRACTION, CRAWLING, LINK DISCOVERY, MACHINE LEARNING, NEWS CLASSIFICATION, LONG SHORT-TERM MEMORY, NEURAL NETWORKS

ВСТУП

Стрімкий зріст інтернет-технологій, а також здешевлення та покращення доступності як і користувацьких пристроїв з доступом у інтернет, так і серверних потужностей для створення власного контенту у мережі. Мільярди тільки проіндексованих пошуковими сервісами сторінок містять дані на будь-яку тематику, і як і об’єм, так і різноманіття цих даних з кожним роком тільки збільшується. Завдяки цьому інтернет став важливим, потужним та всеосяжним джерелом абсолютного різноманіття неструктурованих даних.

Дані (а особливо – видобута з даних істотна інформація) є ключовими у прийнятті рішень, здійснення досліджень, тощо. Коректні, повні дані у достатньому об’ємі дають можливість розуміння досліджуваного предмету, явища, проблеми. Саме тому видобування даних з мережі Інтернет є розвиненою та поширеною практикою як у веденні бізнесу, так і у здійсненні досліджень.

Якісний веб-скрапінг є поширеною та популярною послугою, однак для отримання якісних масових даних з багатьох ресурсів і досі потрібно розробляти спеціалізовані екстрактори даних з конкретних ресурсів або закладати істотні фінансові витрати на розробку таких екстракторів підрядниками які на цьому спеціалізуються або закладати співставні витрати на придбання ліцензій промислових сервісів, які надають API для використання високотехнологічних екстракторів, побудованих на алгоритмах машинного зору та застосовують засекречені алгоритми очистки та перевірки видобутих даних.

Однак, для багатьох досліджень та бізнесів такий підхід є фінансово недосяжним, отже у простих аналітичних алгоритмів є місце для розвитку – як з точки зору дешевизни та простоти розробки та підтримки, так і у невибагливості до умов експлуатації (наявних обчислювальних потужностей. Дана робота присвячена поліпшенню простих алгоритмів та доведенню доцільності їх використання.

В рамках даної роботи був проведений аналіз існуючих методів екстракції даних з ресурсів у мережі інтернет, та розроблено алгоритм видобування новинних матеріалів. Також було розглянуто методи класифікації новинних текстів. Розроблений алгоритм було імплементовано та порівняно з наявним у індустрії програмним забезпеченням за допомогою тренування алгоритмів машинного навчання на зібраних відповідними екстракторами даних, та застосовано результати натренованих алгоритмів для розробки браузерного додатку попередження про неякісні новини.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

У даному розділі розглянуто теоретичні відомості про існуючі методи видобування структурованої інформації з веб-сторінок та методи машинного навчання для автоматичної оцінки ймовірності достовірності статті інтернет-видання на основі лексичної та синтаксичної схожості на статті, опубліковані виданнями з підтвердженою репутацією.

1.2 Веб-скрапінг

1.2.1 Загальні положення про веб-скрапінг

DOM (Document Object Model, об’єктна модель документа) – програмний інтерфейс браузера для роботи з структурою відображеного документу, надаючи можливість для динамічної зміни структури документу, його зовнішнього вигляду та вмісту. Після аналізу завантаженого браузером з серверу документа рушій браузера будує деревоподібне представлення цього документа. Після побудови DOM-дерева скриптовий код, завантажений відкритим у браузері документом або включений в нього отримує доступ до редагування цього дерева.

Дерево складається з множини вузлів, кожен з яких належить до одного з таких типів:

- Документ – є коренем дерева з розширеним відносно інших елементів набором можливостей для взаємодії, та містить у собі весь відображений або схований за допомогою CSS вміст завантаженої сторінки;
- Елемент – є окремим повноцінним елементом дерева. Може бути як і проміжним елементом гілки документа, містячи у собі текст чи інші елементи, так і листком DOM-дерева, відображаючи у структурі документу зображення, елемент форми (поле вводу, прапорець, тощо) або набір стилів або скриптів;

- Атрибут – є параметром елемента DOM-дерева, може регулювати його зовнішній вигляд, поведінку в взаємодіях з користувачем, тощо;
- Текст – є листком DOM-дерева, надає доступ до текстових даних;
- Фрагмент документа – не є вузлом DOM-дерева, та використовується для створення елементів поза DOM-деревом перед приєднанням новоствореного скриптом, завантаженим сторінкою, вмісту сторінки.

API (Application programming interface) – структурована система визначень, яка описує протоколи комунікації, структури даних, способи визову підпрограм, тощо, для формалізованої взаємодії між застосунками, додатками, сервісами, операційною системою, приладами, тощо. В розробці програмних комплексів, які спираються на взаємодію комп'ютерів через мережу Інтернет API як правило представляє собою набір адрес взаємодії, повідомлень та відповідей, таким чином формалізуючи комунікацію між застосунками або пристроями через комп'ютерні мережі. В веб-розробці API як правило визначається на базі можливостей та запитів, визначених у протоколі HTTP. Найчастіше комунікація будується за допомогою передачі повідомлень, які містять у собі структуровані в XML або JSON-документи, але у деяких специфічних випадках застосовуються надбудови, такі як gRPC та GraphQL.

Headless browser – Веб-браузер без користувацького інтерфейсу, який надає API для зовнішніх програмних засобів для завантаження, відображення, автоматичної взаємодії та аналізу веб-сторінок. Застосовуються для скрапінгу, зняття скріншотів, автоматизації тестування, написання ботів, тощо.

Веб-скрапінг, або просто скрапінг – процесс видобування структурованих або неструктурованих даних з інтернет-сторінки, призначених для візуального споживання людиною. Скрапінг множини веб-сторінок називається "краулінг".

Скрапінг-програми можуть працювати за декількома принципами:

- Імітація природньої поведінки людини у реальному або headless браузері, включно з натуральною динамічною поведінкою курсора та інших засобів взаємодії з веб-сторінкою

- Аналіз DOM-дерева – застосування headless-браузера для завантаження сторінки, пошуку необхідних для аналізу та вивантаження DOM-елементів та доступу до їх вмісту.
- Ручний парсинг – завантаження документа за протоколом HTTP та самостійне видобування цільових даних з тексту HTML-документа, включаючи пошук тегів, очищення даних, тощо.

В будь-яких скраперах є чотири робочих етапи:

1. Завантаження – на цьому етапі програмний засіб звертається до веб-сервера та отримує від нього HTML-документ сторінки. У випадку використання реального або headless-браузеру до цього етапу також входить завантаження та виконання ініціалізаційного скриптового коду веб-сторінки та очікування дозавантаження та відображення інформації, яка додається ініціалізаційним скриптовим кодом;
2. Пошук – на цьому етапі виявляються необхідні вузли чи секції документа, видаляються зайві або заважаючі виділенню даних;
3. Аналіз – на цьому етапі з знайдених вузлів чи секцій виділяється та очищається необхідна інформація, виконується первинна обробка даних та приведення їх в стан, готовий до відправки;
4. Вивантаження – на цьому етапі дані, вийняті з веб-сторінки валідуються та з ними виконуються подальші дії, як-то відправлення на сервер, зберігання у файл, тощо. Виконується вивантаження сторінки та завершення роботи.

Краулінг – регулярний масовий скрапінг сторінок сайту або набору сайтів для виділення або обчислення з даних сайту цілісної та повної структурованої інформації.

1.2.2 Концепція веб-скрапінгу

Наявність доцільних, коректних і повних даних є однією з ключових потреб бізнесу та досліджень. Дані є основою для прийняття рішень, розуміння розвязуваних проблем та підтвердження наукових та бізнес гіпотез.

Особливо дані є важливими для створення програмних систем класифікації, кластеризації, тощо, на основі алгоритмів машинного навчання, оскільки ті є напряму залежними від повноти та якості наданих для їх тренування даних.

Важливим джерелом даних для вирішення вищезазначених проблем та пошуку відповідей на пов’язані з проблемами питання є інтернет. Сучасний інтернет є величезним постійно зростаючим океаном даних різного виду і на різні теми.

Для автоматизованого збору даних з різних джерел у мережі інтернет використовуються скрапери. За допомогою веб-скраперів збирають та моніторять, наприклад, актуальні в соціумі теми (за допомогою аналізу соціальних мереж та новинних сайтів), контактні дані для пошуку клієнтів, ціни та послуги/товари конкурентів для аналізу ринку, тощо.

Серед великої кількості різноманітних видів програмних застосунків для автоматизованої екстракції структурованих даних з веб-сторінок можна виділити такі основні типи:

- Спеціалізовані скрапери – скрапери, побудовані для видобування даних з конкретного веб-сайту, розраховані на роботу лише з однією DOM-структурою (та, часто, загальною структурою сторінок сайту). Такі скрапери видобувають дуже “чисті дані”, оскільки вони працюють у чітко визначеному середовищі та ідеально під нього підганяються, та можуть бути оптимізовані відповідно до будови сайту. Головним мінусом таких скраперів є їх головний плюс – такий тип вимагає спеціалізованих правил видобування даних для кожного цільового

ресурсу, та часто важко піддається модифікації для роботи з іншими ресурсами, що робить такі скрапери доцільними лише у окремих випадках [3];

- Ресурс-агностичні типові скрапери – скрапери, розраховані на видобування даних з однорідних (“типових”) сайтів, наприклад сайтів з пошуку роботи, новинних сайтів, тощо. Такі скрапери видобувають дані, орієнтуючись на пошук стандартних елементів або базових розмітки, які часто спираються на best practices індустрії розробки веб-сайтів. Такі скрапери балансують між універсальними скраперами та спеціалізованими скраперами, комбінуючи відносну простоту розробки та універсальність (обмежену на один вид типової побудови веб-сторінок з невеликими варіаціями). Також, таким застосункам для автоматизованого видобування інформації притаманна можливість з відносною простотою вносити зміни в програмний код для адаптації скрапера для роботи з тими ресурсами, які не повністю відповідають структурі, на яку орієнтований конкретний скрапер. Відповідно, мінусами такого підходу є те, що в певній предметній області ресурсів, на яку орієнтований програмний продукт трапляються ресурси, які не були побудовані відносно стандартам індустрії та/або мають нетипову для сайтів цієї предметної області структуру, та те, що, як правило, у скраперах “перехідного” типу відсутній етап очистки даних та/або дерева сайту, тому існує імовірність забруднення даних рекламними матеріалами, інтерактивними елементами, тощо;
- Ресурс-агностичні універсальні скрапери – скрапери побудовані за допомогою алгоритмів, які виконують tree pruning та pattern matching [1] або computer vision [4] для класифікації сторінки та її частин і подальшого видобування структурованої інформації. Оскільки програмні комплекси, які входять в цю категорію, є наукоємними та використовують ідейно та обчислювально складні алгоритми, такі

скрапери є складними в розробці та підтримці, вимагають більших обчислювальних потужностей та складнішої архітектури для забезпечення масштабованості системи. Цей вид скраперів як правило розробляється як внутрішній продукт компаніями, які вимагають великої кількості різноманітних даних, або, що частіше, окремими бізнесами, які надають послуги з видобування та очистки значних об’ємів даних.

1.2.3 Існуючі рішення у сфері веб-скрапінгу

Відомі скрапінг-продукти:

- grabz.it – надає ряд послуг, таких як: знімки веб-сайту, архівація, побудова коротких прев’ю посилань, тощо. Також дає можливість будувати спеціалізовані скрапери, як на серверах користувача, так і в хмарному середовищі grabz за допомогою HTTP-запитів до API grabz;
- webscraper.io – інструмент для побудови спеціалізованих скраперів без коду, у інтерактивному інтерфейсі, дозволяє будувати власні структури даних засновуючись на потребах бізнесу. Має можливість застосування як у вигляді браузерного розширення, так і на власних серверах чи в хмарному середовищі, виконуючи HTTP-запити до API, яке надається сервісом webscraper;
- parsehub.com – аналог webscraper.io, спеціально пристосований для видобування великих об’ємів даних;
- diffbot.com – ресурсо- та областe-агностичний продукт, який застосовуючи алгоритми машинного навчання, будує графи знань з наданих наборів веб-сторінок та інструментарій для аналізу побудованих графів знань. Важливими особливостями роботи даного сервісує властивість мімікрувати під природньо-людську поведінку для обману ресурсів, які знаходяться під прицілом сервісу;

- scrapinghub.com – Надає API для користування спеціальними ресурсо-агностичними, але специфічними для конкретної предметної області скраперами. Якість та чистота даних при збереженні ресурсо-агностичності хмарних скраперів, які надаються сервісом scrapinghub досягається за допомогою налаштовуваних аналітичних алгоритмів, які визначають структури аналізованої сторінки та багатокрокового автоматичного, автоматизованого та ручного процесів контролю якості даних. [5]

1.3 Класифікація дезінформативних новин

1.3.1 Загальні положення про класифікацію дезінформативних новин

1.3.1.1 Загальні визначення

Дезінформація – вид інформаційної атаки на конкретну людину, організацію чи країну. Відноситься до категорії психологічних впливів. Спосіб дезінформації полягає у поданні реципієнту інформації таких даних, які подають йому неправдиву інформацію, створюючи викривлену реальність у розумінні та інформаційному полі частки суспільства щодо реального стану справ у певній області. Маніпулятивна та свідомо хибна інформація використовується для досягнення різних цілей, наприклад:

- комерційних (погіршення репутації конкурента);
- пропагандистських (маніпуляція соціальним настроєм для досягнення політичних цілей);
- військових (введення військового керівництва, середньої офіцерської ланки чи солдатів противника в оману).

Фейк – підвид дезінформації, навмисна і відверто хибна інформація

Маніпуляція – дезінформативна техника, яка полягає у свідомому або несвідомому (для жертв дезінформативних технік) поданні видозміненої правди (загальна інформація про подію збережена, ролі учасників видозмінені для висвітлення окремих особистостей, організацій чи країн у позитивному чи негативному світлі) за допомогою точкової брехні та емоційно забарвленої лексики.

1.3.1.2 Визначення, специфічні до сфери обробки природних мов

Токенізація – перетворення тексту як послідовного набору символів на послідовний набір значущих одиниць тексту, при чому на один токен – значиму одиницю тексту – може припадати від невеликої частини одного слова до

декількох слів (в залежності від характеристик аналізованої мови, контексту речення, тощо)

TF-IDF (term frequency-inverse document frequency) – статистичний показник у обчислювальній лінгвістиці та обробці природних мов. TF – частота вживання слова в конкретному документі. IDF – логарифм від загальної кількості документів, розділеної на кількість документів у яких трапляється вищезазначене слово, та позначає рівень уживаності слова в наявному корпусі документів. Таким чином, множення TF на IDF дає можливість оцінити важливість слова у документі відносно до важливості слова у корпусі, частиною якого цей документ є, таким чином даючи метрику релевантності терміну.

Шумові слова (стоп-слова) – слова, які не містять інформації і є зайвими для алгоритмів пошуку чи аналізу тексту. Вони поділяються на загальні (цифри, знаки, прийменники, слова які застосовуються занадто часто у масиві на якому здійснюється аналіз, тощо), та залежні (такі, які не містять інформації у конкретному прикладі аналізованого тексту).

Наївний байєсів класифікатор – класифікатор на основі теорії ймовірності. Для визначення ймовірності входження тестованого елемента до класу використовує теорему Байєса, базуючись на наївному припущенні незалежності параметрів елемента.

Перцептрон – вид найпростіших нейронних мереж, які складаються з трьох видів нейронів: рецептори, асоціативні нейрони та реагуючі нейрони. Рецептори є вхідними нейронами мережі, які збирають вхідні дані. Кожен рецептор в вхідному прошарку нейронної мережі з’єднаний з лише одним нейроном наступного шару, який складається з асоціативних нейронів. Асоціативні прошарки є прихованими прошарками нейронної мережі та відповідають за обробку вхідної інформації. Кожен нейрон асоціативного прошарку пов’язаний з кожним нейроном попереднього та наступного прошарків. Після одного або декількох асоціативних прошарків мережі знаходиться прошарок реагуючих нейронів, які є виходами нейронної мережі, та виводять інформацію назовні.

Пасивно-агресивний класифікатор – інтерфейсно подібний перцептрону класифікатор, пристосований для роботи з великими масивами потокових даних. Його особливістю є специфічний алгоритм навчання, який забезпечує швидке навчання та можливість навчання або додаткового навчання на потокових даних. Назва походить від подвійної поведінки під час навчання: Пасивна (якщо передбачення правильне – модель не треба змінювати, оскільки такі дані не є істотними причинами для зміни моделі), Агресивна (якщо передбачення хибне – модель коригується оскільки такі дані є істотними для її роботи).

Рекурентні нейронні мережі – клас алгоритмів машинного навчання, у якому дозволяється з'єднання глибших прошарків мережі з ближчими до поверхні, що формує орієнтований в часі граф та дозволяє штучній нейронній мережі мати внутрішній стан, та змінювати свою подальшу поведінку в залежності від попередніх вхідних даних.

LSTM – різновид рекурентних нейронних мереж, які містять LSTM-вузли. LSTM-вузол є рекурентним вузлом, у якому не застосовується функція активації для рекурентних складових вузла, що дозволяє нейронній мережі мати можливість запам'ятовувати дані як на короткий, так і на довгий проміжок часу. Такі вузли як правило складаються з чотирьох складових –- "клітина", яка відповідає за пам'ять, і три клапани – вхідний, вихідний та стираючий пам'ять.

Word2Vec – підхід та комплекс нейронних мереж для вивчення словесних асоціацій з великих об'ємів даних. Word2Vec-мережі є простими двошаровими мережами, націленими на конструювання лінгвістичних контекстів слів. Ці мережі навчаються на величезних корпусах для створення багатовимірних (декілька сотень) векторних просторів, в яких кожному слову призначається певна точка в цьому просторі, а схожі за контекстом використання слова отримують геометрично близькі позиції.

1.3.2 Концепція класифікації дезінформативних новин

На даний момент, дезінформація та “фейкові новини” є однією з найпоширеніших проблем українського та світового суспільства. Дезінформацію широко використовують у якості інструменту для досягнення цілей через те, що більшість людей не є достатньо компетентними в темах, які висвітлюються в новинах та у соціальних мережах, та занадто часто не займаються перевіркою знайденої інформації. Погіршує проблему те, що велика кількість користувачів соціальних мереж дізнаються вагому частину новин саме з них, у вигляді постів у групах, на які підписані, та репостів їх друзів. Навіть досвідчені журналісти з солідним досвідом іноді пропускають у видання фейкові новини, та рідко визнають свої помилки.

Дезінформативні тексти можуть містити як відверто маніпулятивний стиль викладення, так і бути “прихованими” за шаром правдивої інформації, тому, на жаль, на даний момент найпоширенішим інструментом у визначенні дезінформативних новин є ретельний ручний факт-чекінг. Головною причиною цього є те, що “якісно” зроблені дезінформуючі новини виглядають як новини, які дійсно могли б бути справжніми і подаються у серйозному журналістському стилі, який найчастіше використовується відомими виданнями з заслуженою роками репутацією та жорсткими і вимогливими журналістськими стандартами видобування, перевірки та викладення інформації.

Нейронні мережі та інші алгоритми машинного навчання як правило спираються на лексичний та семантичний аналіз текстів, тому легко пропускають такі новини, оскільки справжнє розуміння сенсу тексту та визначення достовірності викладеної в цьому тексті інформації вимагає від штучного інтелекту розуміння соціальних та інших контекстів аналізованої ним інформації, а також наявність у алгоритму *усвідомлення* очевидних та повсякденних для людини речей, тощо. Аналогічною визначенню дезінформації у тексті задачею є, наприклад, точний машинний переклад, який вимагає від

штучного інтелекту *розуміння* підтексту, або, мовою психологічної теорії трансактного аналізу “прихованих трансакцій”. Підтекст, або прихована трансакція – це один з видів комунікації людей, в якому є два канали передачі інформації – прямий (вербальний) та прихований (невербальний). Поширеним прикладом прихованих трансакцій є прислів’я, наприклад “В чужому оці порошинку бачить, а в своєму й сучка недобачає”. Задача розуміння невербальних та відомих і усвідомлених людьми підтекстів наразі вважається AI-повною задачею, тобто задачею яка вимагає створення штучного інтелекту, подібного до людини та рівного за знаннями та можливостями усвідомлювати інформацію. Відповідно, це робить задачі визначення недостовірної інформації та задачі машинного перекладу також AI-повними.

Однак, існування численних програмних продуктів та сервісів, які займаються машинним перекладом і таким чином спрощують роботу перекладачам та спрощують іншим людям розуміння текстів, написаних невідомою для них мовою або допомагають їм комунікувати з людьми, які не розуміють їх мови та іншим чином полегшують міжмовну комунікацію подає приклад того що навіть у AI-повних задачах вирішення аналітичними алгоритмами або алгоритмами машинного навчання спрощених та обмежених підзадач в цьому полі може бути якщо не достатнім у деяких випадках рішенням, то хоча б полегшувати це завдання.

Саме тому вирішення задачі класифікації тексту як *схожий* на дезінформацію чи *схожий* на правду є актуальною темою, на яку звертають увагу багато науковців та інженерів.

1.3.3 Існуючі рішення в сфері класифікації дезінформативних новин

Серед проектів, які займаються боротьбою з дезінформацією, маніпулятивними новинами та іншими пов’язаними явищами було знайдено чотири такі проекти, які використовують машинне навчання у своїй роботі:

- botsentinel.com – відкрита організація, яка надає безкоштовний доступ до бази так званих “ботів” у соцмережі Twitter та даних про те, на які теми вони пишуть. Машинне навчання використовується для аналізу текстів у соцмережі та кваліфікації їх як заслуговуючих чи не заслуговуючих довіри, але результати роботи перевіряються вручну волонтерами для забезпечення якості даних;
- botcheck.me – мертвий проект, який являє собою браузерний додаток для ідентифікації ботів, що розповсюджують дезінформацію або маніпулятивні новини за допомогою машинного навчання у соцмережі Twitter;
- botometer.osome.iu.edu – веб-сайт, який використовує алгоритми машинного навчання та зовнішні ресурси для збору даних для ідентифікації несправжніх акаунтів у соцмережі Twitter. Для оцінки за шкалою від 1 до 5 проект застосовує кілька алгоритмів машинного навчання, які працюють з текстами, викладеними ботом, його активністю у часі та загальною інформацією (фото, ім’я, опис, тощо)
- trustedtimes.org – браузерний додаток для аналізу текстів на новинних сайтах. Використовує машинне навчання для визначення ступені “правдоподібності” новини, ідентифікації політичної позиції тексту (щодо яких політиків стаття створює негативну/позитивну/нейтральну картину);
- fgz.texty.org – браузерний додаток, який нотифікує користувача якщо той відвідує сайти, які є ненадійними джерелами, містять маніпуляції, фейки чи “джинсу” (Замовні тексти, зазвичай політичного напряму, спрямовані на знищення чи навпаки поліпшення репутації компанії, персони, політичного угрупування, тощо). Використовують машинне навчання, користувацькі мітки та дані Інституту Масової Інформації. Єдиний проект з усіх знайдених який вміє працювати з українською та російською мовами;

У автоматизованій оцінці текстів на імовірність включення дезінформації, фейків, тощо часто використовують наступні методи для підготовки даних та перетворення їх у форму, прийнятну для використання у тренуванні та класифікації за допомогою алгоритмів машинного навчання:

- Використання TFIDF для побудови загальної характеристики слів, використаних у тексті відносно загального масиву слів у новинному лексиконі, та подальша векторизація отриманих даних для тренування алгоритму машинного навчання. Недоліками цього підходу є необхідність в обчисленні TFIDF кожного слова відносно усього корпусу документів, що ставить перед розробником системи необхідність у оптимізації процесу та кешуванню значень, а також постійного оновлення корпусу документів у обчисленні TFIDF новими документами, та відсутність можливості обробки фраз у цілому, а не слів, з яких вони складаються. Плюсом є ефективність у пошуку емоційно забарвленого лексикону, який часто є маркером маніпуляційних та фейкових новин;
- Використання токенізаторів для побудови лексичного корпусу, виділення значущих слів та подальшої векторизації токенізованих даних для тренування алгоритму машинного навчання. Недоліками є необхідність у ретельній фільтрації шумових слів та необхідність алгоритму машинного навчання самостійно виділяти значимі слова, на відміну від TFIDF. Плюсом же цього підходу є можливість подавати на вхід, наприклад, LSTM-мережі або іншої мережі з пам’яттю тексту послідовно, маючи таким чином можливість розглядати фрази та речення не тільки послівно, а і в цілому, реалізуючи наявність внутрішнього стану мережі;
- Використання Word2Vec як частини нейронної мережі або передтренування його як окремої мережі та використання у якості проміжного етапу для створення ембедінгів. Недоліком цього підходу є

необхідність у алгоритмі тренування, специфічному до Word2Vec та необхідність у тренуванні на величезних об’ємах даних для повноцінної реалістації властивості Word2Vec підходу розпізнавати контекстні синоніми та видавати їх як геометрично близькі у векторному просторі.

Для проведення класифікації використовуються наступні методи:

- Наївний байєсів класифікатор – простий класифікатор, який легко та швидко навчається і може використовуватись для обчислень на клієнтській стороні через свою швидкодію;
- Пасивно-агресивний класифікатор – класифікатор, який пристосований до швидкої обробки великих масивів даних, тому використовується у обробці великих потокових даних та там, де важлива можливість реагувати швидко, наприклад у потоковій обробці живих даних соцмереж;
- Рекурентні нейронні мережі, у тому числі рекурентні мережі архітектури довгої короткочасної пам'яті, які дають можливість проводити глибокий аналіз тексту, реалізуючи наявність у рекурентних мереж внутрішнього стану, який дозволяє аналізувати фрази та речення в цілому;
- тощо.

1.4 Постановка задачі

1.4.1 Призначення розробки

Розроблений екстрактор призначений до використання для видобування інформації з новинних веб-ресусів для застосування алгоритмами машинного навчання.

1.4.2 Цілі та задачі розробки

Розроблений програмний продукт має надавати прозорий та зручний спосіб видобування заголовків та текстів новин з довільних новинних ресурсів, та у якості результату видавати JSON-файл з результатами видобування. Також екстрактор має обробляти файл robots.txt цільового ресурсу для дотримання «контракту» доброчесного користування ресурсом, та мати можливість встановлювати затримку між окремими запитами. Як опцію екстрактор має приймати також користувацькі фільтри посилань, якими користувач може визначити сторінки, які екстрактор ігноруватиме.

1.5 Висновок до розділу

У даному розділі було викладено аналіз двох предметних областей з точки зору поставленої задачі, описано необхідну термінологію та необхідні загальні дані.

У першій частині розділу було проаналізовано існуючі підходи до екстракції структурованих даних з веб-сторінок та їх множин та наведено і розглянуто існуючі програмні засоби та сервіси, які вирішують цю чи аналогічні задачі за допомогою різних підходів.

У другій частині розділу було викладено концептуальні та філософські засади і доцільність використання у сфері визначення та класифікації новин як маніпулятивні, дезінформаційні або інформаційно забарвлені програмних засобів загалом і, зокрема, алгоритмів машинного навчання. Також було викладено і описано існуючі підходи у попередній підготовці та обробці текстових даних для використання у алгоритмах машинного навчання, розібрано типові існуючі підходи з класичного машинного навчання та нейронних мереж і вказано існуючі рішення в цій сфері, включно з описом їх специфік.

В кінці розділу було сформульоване коротке завдання на розробку та описано цільове призначення майбутнього продукту.

2 МЕТОДИ ЖАДІБНОГО КРАУЛІНГУ ТА МАШИННОГО АНАЛІЗУ ОТРИМАНИХ ТЕКСТІВ

2.1 Алгоритм жадібного краулінгу

Для тестування екстрактора було обрано чотири ресурси з підтвердженою позитивною репутацією і низькою кількістю маніпулятивних новин за рейтингами Інституту Масової Інформації, тощо. Також було обрано п’ять новинних ресурсів з підтвердженою негативною репутацією або невідповідністю стандартам журналістики[1]. Повний список обраних видань наведено нижче:

- bbc.com/{‘ukrainian’/’russian’};
- novoross.info;
- hyser.com.ua;
- ua3.info;
- meduza.io;
- liga.net;
- 40ka.info;
- radiosvoboda.org;
- prefiksblog.co.ua.

Усі з вищезазначеної вибірки інформаційно-новинних ресурсів мали одноманітну DOM-структуру релевантно до дослідження даної магістерської роботи:

- Заголовок статті має тег h1 чи h2;
- Текст статті має теги p, або інші теги всередині тегу article;
- Використовуються теги a для посилань на інші сторінки.

Оскільки проаналізовані новинні ресурси мали одноманітні структури з декількома мінорними варіаціями, що може бути продиктоване використанням семантичної розмітки та індустріальних стандартів веб-розробки з метою

[1] За оцінкою авторів роботи, керуючись довідкою [18]

покращення ранжування матеріалів новинного ресурсу, тому спираючись на інформацію, отриману в процесі підготовки даних та написання першого розділу цієї роботи для реалізації було обрано проміжний збалансований метод побудови скрапінгового алгоритму.

Зважаючи на вищезазначені дані реалізація універсального скрапера з використанням обчислювально важких та ресурсовимогливих алгоритмів, таких як tree pruning або visual content detection для використання з новинними ресурсами була розцінена як недоцільна та неоптимальна для використання у задачі видобування новинних матеріалів для їх подальшої класифікації.

Базовий процес винайденого алгоритму жадібного краулінгу веб-сайту виглядає таким чином:

```
Отримати цільовий веб-ресурс та частини ресурсу, які не слід завантажувати
Просканувати robots.txt, та додати описані там заборони до списку заборон
Створити пустий Map вже переглянутих сторінок
Створити Set “наступних сторінок”
Додати в наступні сторінки першу сторінку
Створити інстанс headless-браузеру
Для кожної сторінки у сеті наступних сторінок:
    Завантажити сторінку
    Якщо мова сторінки не входить до мов, які визначені для скрапінгу:
зберегти поточну сторінку як переглянуту перейти до наступної сторінки в списку
    Для кожного елементу “a” на сторінці:
        Видобути посилання тегу
        Якщо посилання не включає домену:
```

Додати домен до посилання

Зберегти посилання в тимчасовий масив

Для кожного елементу “h1” та “h2” на сторінці:

Видобути текст тегу

Обрати найдовший текст з попереднього кроку

Якщо найдовший заголовок більший за HEADER_THRESHOLD символів вважати сторінку імовірною статтею

Для кожного елементу “p” та елементів усередині елементу “article”:

Прибрати з дітей тегу елементи виду “button”, “a”, “script”, “style”

Видобути суцільний текст вищезазначених елементів

Якщо суцільний текст більший за ARTICLE_THRESHOLD:

Зберегти заголовок та текст статті, позначити сторінку як переглянуту.

Інакше

Позначити сторінку як переглянуту

Для кожного посилання на сторінці, якщо воно

є посиланням на шукомому домені

ще не переглянуте

ще не записане в сторінки до перегляду

не включає частин посилання, які треба виключати

є посиланням у протоколі HTTP:

зберегти посилання у посилання до перегляду

Закрити headless-браузер

Реалізація даного алгоритму на мові програмування ECMAScript з застосуванням інструменту роботи з headless-браузером Pupetteer наведена в додатку X до даної магістерської роботи.

За результатами експериментів зі скрапінгу сайтів, зазначених у списку у початку даного розділу було визначено що вищезазначена реалізація даного алгоритму працює надійно і стабільно, набираючи істотний масив даних за допомогою “жадібного” аналізу всіх посилань наявних на усіх оброблюваних сторінках.

Недоліком цього підходу та цього алгоритму є те, що на деяких новинних ресурсах алгоритм має властивість не відділяти технічну інформацію ресурсу від тексту статті, через що у кінці чи початку статтей з таких ресурсів містяться зайві дані, такі як дата публікації, автор, рекламні дані чи пропозиція прокоментувати чи підписатись на даний ресурс. Оскільки такі помилки видобування та структурування інформації розміщеної на сторінці є стабільними і передбачуваними у межах кожного такого ресурсу цей недолік компенсується постобробкою методом вилучення стабільно протікаючих у текст статті речень, фраз, символьних комбінацій, тощо.

2.2 Алгоритм підготовки даних

Для забезпечення якості результатів тренування алгоритмів машинного навчання було розроблено алгоритм підготовки даних, задачею якого є створення та наповнення файлу навчальних даних з файлів результатів скрапінгу. Також алгоритм виконує первинну обробку тексту видобутих статей, приводячи їх до одноманітного регуляризованого формату. Останнім кроком трансформації є перевірка слів на входження в список шумових слів (російська та українська). Фільтрація шумових слів є важливим елементом ефективності алгоритмів класифікації та категоризації текстів живими мовами, та широко використовується у машинній обробці натуральних мов [10].

Нижче приведено опис послідовності дій, які виконуються під час підготовки даних трансформатором:

```
Відкрити файл з списком шумових слів
Створити Set з списку шумових слів
Для кожного файлу у списку файлів результатів скрапінгу:
    Відкрити файл
    Для кожного запису в файлі:
        Якщо запис не є статтєю:
            Пропустити
        Присвоїти статті маркування 0 чи 1 відповідно до
        її джерела (ненадійне/надійне)
        Видалити джерело з статті
        Залишити в статті тільки буквенні символи та
        пробіли
        Видалити повторні пробіли та знаки переносу рядка
        Привести статтю до нижнього регістру
        Розбити статтю на окремі слова
        Для кожного слова в статті:
            Якщо слово є в списку шумових слів:
                Видалити слово з статті
        Зібрати статтю з залишкових слів
    Зберегти результати обробки файлу в вихіний файл
Перемішати записи в вихідному файлі за допомогою варіації
методу Фішера-Єйтса
```

Даний алгоритм було реалізовано на мові `ECMAScript` для платформи `Node.js`. Виділення та видалення небуквенних та інших символів було виконано за допомогою регулярних виразів. Реалізація алгоритму тасування Фішера-Єйтса була запозичена з бібліотеки реалізацій типових функцій `lodash`.

2.3 Алгоритм TF-IDF-векторизації

TF-IDF тексту у загальному сенсі є набором значень «важливостей» конкретних слів у контексті тексту відносно контексту корпусу документів.

Першою частиною TFIDF є частота слова (term frequency) у документі, яка обчислюється таким чином:

$$TF_i = \frac{n_i}{\sum_k n_k},$$

де TF_i є частотою слова t_i, n_i є кількістю входжень слова у документі, $\sum_k n_k$ є загальною кількістю слів у документі.

Другою частиною TF-IDF є інверсна частота документу – інверс частоти входження слова у корпусі, яка обчислюється таким чином:

$$IDF_i = \log \frac{|D|}{|(t_i \in d_i)|},$$

Де IDF_i є інверсною частотою документу, $|D|$ є об'ємом корпусу, $|(t_i \in d_i)|$ є часткою корпусу, яка містить слово t_i. Варто зауважити, що основа логарифму у цій формулі не має значення доки вона є однаковою у межах дослідження. Зміна ж основи призводить до зміни ваги кожного слова на постійний множник.

Добутком частин TF та IDF є показник TF-IDF:

$$TFIDF_i = TF_i \cdot IDF_i.$$

Оскільки у цій роботі TF-IDF-метрика використовується не для оцінки статичного корпусу текстових документів, а як один з кроків у класифікації даних, які надходять динамічно – для досягнення прийнятної швидкодії результуючого програмного забезпечення формується лексично репрезентативний (відносно текстів, які будуть підлягати аналізу) корпус документів, і обчислюється IDF словника такого корпусу.

Це відбувається за таким принциповим імперативним алгоритмом:

```
Для кожного слова в кожному документі корпусу:
    Якщо слово не входить в словник:
        Додати слово в словник та призначити йому індекс
Замінити слова в документах на відповідні індекс
Створити вектор входжень розміром, рівним словнику, заповнений 0
Послідовно для кожного індексу в кожному документі корпусу:
    За індексом збільшити на 1 значення в векторі входжень
Для кожного елементу вектору входжень:
    Обчислити дільник кількості документів корпусу на значення елементу
    Зберегти в елемент логарифм значення вище.
Зберегти словник та вектор IDF
```

Для обчислення TF-IDF та підготовки вхідної матриці алгоритму машинного навчання використовується такий алгоритм з використанням значень, передобчислених алгоритмом зазначеним вище:

```
Завантажити словник та вектор IDF
Створити масив-результат з елементами [індекс в словнику, лічильник входжень]
Для кожного слова в документі:
    Якщо індекс слова в словнику є в масиві результаті:
        Збільшити лічильник входжень слова на 1
    Інакше:
        Додати [індекс, 1] в масив
Для кожного елементу в масиві входження:
    Поділити лічильник входжень слова на кількість слів у масиві входження, домножити на значення IDF взяте по індексу слова в словнику
```

```
    Замінити лічильник входжень на значення, обчислене в
попередньому кроці.
```

2.4 Алгоритм токенізації

Альтернативним методом підготовки сирого тексту є токенізація. У цьому методі ми навчаємо токенізатор найчастішим словам збираючи словник, та за допомогою цього словника обертаємо набір слів тексту на вектор індексів цих слів у словнику.

Внутрішній вигляд словника представлено таким чином:

$$[\text{слово1}, \text{слово2}, \text{слово3}, \dots]$$

є словником відповідності індексу слову,

$$\{\text{слово1}: 1, \text{слово2}: 2, \text{слово3}: 3, \dots\}$$

Є словником відповідності слова індексу.

Словник наповнюється методом аналізу доступного корпусу тестів та обрізання до максимуму необхідних:

```
Створити словник входжень слів
Послідовно для кожного слова в кожному документі корпусу:
    Якщо слово є в словнику входжень:
        Збільшити кількість входжень на 1
    Інакше:
        Поставити кількість входжень слова як 1
Відсортувати слова за входженнями
Для кожного слова в n найвживаніших слів:
    Призначити слову індекс в словнику відповідності
індексів словам
    Призначити індексу слово в словнику відповідності слів
індексам
```

Оскільки даний метод використовуватиметься для підготовки даних для використання у нейронній мережі такий метод має видавати вектор сталого

розміру, і, відповідно, стандартний метод токенізації тексту було модифіковано. В стадію підготовки токенізатора було додано етап обчислення середньої довжини тексту у корпусі як сталого розміру вхідного вектору нейронної мережі. Алгоритм підготовки тексту з участю токенізації та відповідно змін описаних вище виглядає таким чином:

```
Для кожного слова в документі:
    Якщо слово наявне в словнику:
        Замінити слово на індекс з словника
    Інакше:
        Замінити слово на 0
Якщо довжина токенізованого вектору документа менша за середню довжину документу корпусу:
    Розширити від початку вектор на кількість нулів, яка відповідає різниці довжини документа та середньої довжини документів корпусу
Інакше:
    Обрізати вектор до відповідності середній довжині документів корпусу
```

Таким чином, за допомогою доповненої токенізації досягається рівномірний, передбачуваний та вирівняний вхідний вектор нейронної мережі.

2.5 Пасивно-Агресивний класифікатор

Пасивно-агресивний класифікатор належить до сімейства бінарних класифікаторів послідовного навчання. Ідейно він є подібним до методу опорних векторів (SVM, support vector machines). Пасивно-агресивний класифікатор шукає гіперплощину розділення простору вхідних даних на дві половини. Розділення аналізованого елемента пропорційне відстані до гіперплощини, а похибки можуть бути зкореговані розділенням гіперплощини. Оновлення класифікатора підпорядковується пасивного оновлення та агресивного оновлення – пасивного при співпадінні передбачення з реальністю та агресивного при розбіжності.

Алгоритм навчання пасивно-агресивного класифікатора виглядає таким чином: [11]

Отримати параметер агресивності $C > 0$

Ініціалізувати $w_1 = (0, \dots, 0)$

Для $t = 1, 2, \dots$

Отримати вхідні дані $x_t \in \mathbb{R}^n$

Передбачити $\hat{y}_t = sign(w_t \cdot x_t)$

Отримати правильну мітку вхідних даних $y_t \in \{-1, +1\}$

Отримати значення функції втрат $l_t = \max\{0, 1 - y_t(w_t \cdot x_t)\}$

Отримати $\tau_t = \frac{l_t}{||x_t||^2 + \frac{1}{2C}}$

Оновити модель $w_{t+1} = w_t + \tau_t y_t x_t$

Де w_t є вагою вектора на етапі t, y_t є розділенням на етапі t.

w_{t+1} призначається як проекція w_t на півпросторі векторів з втратою у 0. Таким чином, $w_{t+1} = w_t$ коли $l_t = 0$. На етапах з $l_t > 0$ алгоритм видозмінює w_{t+1} агресивно для забезпечення обмеження $l(w_{t+1}; (x_t y_t)) = 0$ незалежно від істотності необхідної зміни. Через таку поведінку алгоритм і було названо Пасивно-Агресивним.

2.6 Двустороння Довга Короткочасна Пам’ять

2.6.1 Архітектура ДКЧП-мереж

Архітектура довгої короткочасної пам’яті була винайдена у 1997 році та представлена у [12].

До винаходу цієї архітектури призвів аналіз проходу похибки в існуючих на той момент рекурентних нейронних мережах, який показав що довга пам’ять мережі була недоступна в тогочасних архітектурах, оскільки зворотньо поширена похибка має властивість експоненційно зменшуватись або збільшуватись.

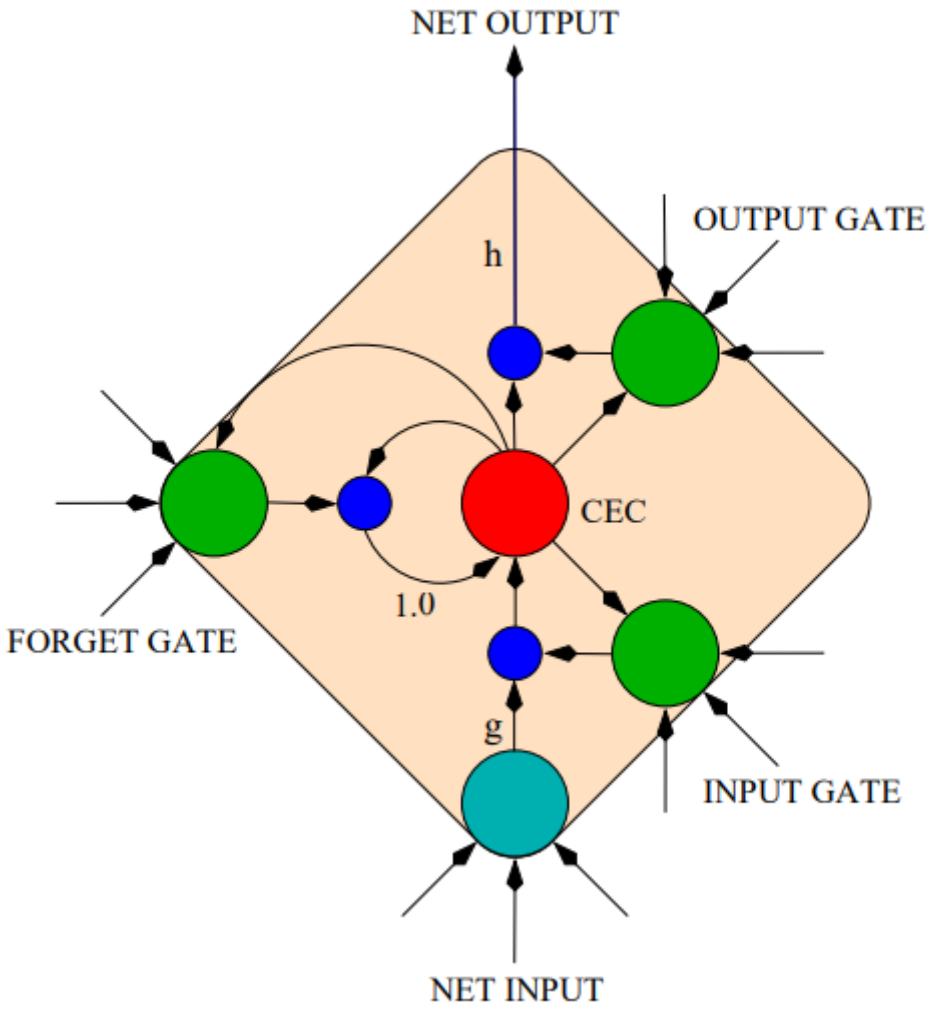

Рисунок 2.1 – ДКЧП-блок з однією клітиною

ДКЧП-прошарок створений з множини рекурентно зв’язаних блоків, так званих «блоків пам’яті», схожі на диференційовані версії оперативної пам’ятті сучасного комп’ютеру. Кожен такий блок складається з одного або декількох рекурентно поєднаних блоків пам’яті та трьох елементів множення – воріт входу, виходу та забуття. Мережа може працювати з клітинами тільки через ворота, які є аналогами операцій запису, читання та скидання пам’яті. Таким чином, вхід до клітини мережі множиться на активацію вхідних воріт, вихід клітини в мережу множиться на активацію, а попередні значення клітини множаться на значення воріт забуття.

На рис. 1 зображено ДКЧП-блок з однієї клітини. Внутрішній стан клітини зберігається рекурентним з’єднанням вагою 1. Три ворота збирають активації ззовні та зсередини блоку, та керують клітиною за допомогою елементів множення (синіх кіл). Вхідні та вихідні ворота масштабують вхід та вихід клітини, а ворота забуття масштабують внутрішній стан, наприклад скидаючи його в 0 (провокуючи скидання внутрішнього стану). Стискаючі функції h та g застосовуються в означених місцях.

2.6.2 Обчислення рекурентних мереж

Відмінність ДКЧП-моделей від звичайних рекурентних моделей яскраво прослідковується у математичному вираженні цих нейронних мереж.

Нехай $\{x_1, \dots, x_n\}$ є ембедінгом токенізованого тексту. Тоді результатом роботи рекурентної нейронної мережі є вихідний вектор y_t для кожного токена x_t. Результат обчислюється повторенням наступних рівнянь від $t = 1$ до $t = n$:

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

$$y_t = W_{hy}h_t b_y,$$

де h_t є прихованою послідовністю векторів, W є матриці вагів (наприклад, W_{xh} є матрицею вагів, які поєднують вхідний та прихований прошарки), b позначає вектори упередженості та H є функцією активації прихованого прошарку [13]. За допомогою векторів h рекурентні моделі і досягають ефекту

пам’ятті та впливу попередніх контекстів на поточні. Через проблему “зникаючого градієнта” стандартні рекурентні нейронні мережі не можуть використовувати всю вхідну історію [14].

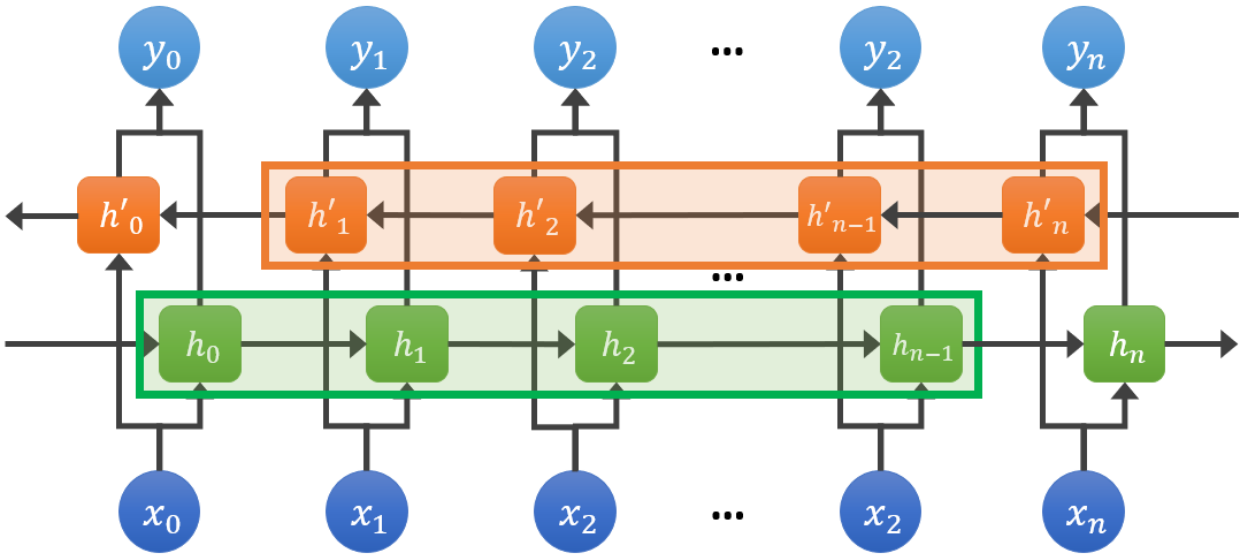

Рисунок 2.2 – Розгорнений у часі нейрон рекурентного прошарку нейронної мережі

2.6.3 Обчислення ДКЧП-мереж

Описані вище принципи роботи нейронної мережі довгої короткострокової пам’ятті (клітини пам’ятті та ворота роботи з нею) застосовані до рекурентної нейронної мережі трансформують її в алгоритм, який може бути описаний таким чином:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$
$$c_t = f_t c_{t-1} + i_t \sigma \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o)$$
$$h_t = o_t \tanh c_t$$

Де σ є логістичною сігмоїд-функцією, i, f, o, та c є векторами вхідних воріт, воріт забуття, вихідних воріт та активації клітини. b є вивченими векторами упередженості.

Однією з проблем як звичайних рекурентних нейронних мереж, так і мереж довгої короткочасної пам’ятті є урахування лише попередніх контекстів.

Живим мовам властиві двусторонні зв’язки у реченнях, який виражається, наприклад, у смисловому впливі слів на початку речення на слова в середині, так і вплив слів у кінці речення на слова в початку, тощо. Такі смислові впливи є невід’ємною частиною органічності походження багатьох мов.

Як приклад можна взяти такі два вигаданих речення:

1. Верховна Рада України прийняла рішення про примусове виселення російськомовних національних меншинств.
2. Верховна Рада України стала місцем мітингу підприємців проти підписання змін до трудового кодексу України.

Очевидним є різна смислова вага словосполучення «Верховна Рада України». Для роботи з такого плану проблемами нейронним мережам, відповідно, необхідне знання не тільки про попередній контекст, а і про майбутній. У контексті нейронних мереж довгої короткочасної пам'ятті для розв'язання цієї проблеми використовуються двусторонні ДКЧП-мережі [15].

Така архітектура нейронної мережи має можливість розв’язання як попереднього контексту, так і наступного контексту конкретного слова.

2.6.4 Обчислення двусторонніх ДКЧП-мереж

Двустороння ДКЧП-модель відрізніється від звичайної тим, що має два окремих прихованих прошарки: перший обчислює пряму послідовність $\overrightarrow{h_t}$, а другий – обернену $\overleftarrow{h_t}$. Після обчислень послідовностей відбувається комбінація $\overrightarrow{h_t}$ та $\overleftarrow{h_t}$ для отримання виходу y_t. Допустимо що h є ДКЧП-блоками, тоді двустороння ДКЧП обчислюється такими функціями:

$$\overrightarrow{h_t} = H(W_{x\vec{h}}x_t + W_{\vec{h}\vec{h}}\vec{h}_{t-1} + b_{\vec{h}})$$
$$\overleftarrow{h_t} = H(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t-1} + b_{\overleftarrow{h}})$$
$$y_t = W_{\vec{h}y}\vec{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y$$

2.6.5 Ембедінги та їх обчислення

Токенізовані тексти мають вигляд цілочислених векторів, у яких кожному слову тексту відповідає унікальне число. Оскільки багато слів можуть мати різні сенси в різних контекстах, або навпаки – декілька слів можуть мати однаковий сенс осмисленим є наявність тренованого прошарку нейронної мережі, який би за вивченим словником виконував трансформацію слів у вектори, які характеризують ці слова відповідно її навченому досвіду. Для цього використовуються Embedding-шари.

Ембедінг-шар трансформує вхідний вектор токенізованого тексту на матрицю, де кожному слову відповідає вектор у n-мірному просторі, який допомагає подальшим прошаркам нейронної мережі характеризувати це слово. По суті ембедінг-шар вивчає lookup-матрицю, в якій кожному слову в словнику відповідає вищезазначений вектор, та під час роботи мережі простим матричним множенням виконує трансформацію.

Таким чином, нейронна мережа з використанням ембедінг шару може наблизитись до розуміння не просто комбінацій слів, а комбінацій сенсів слів (звісно, у тому вигляді, який доступній нейронній мережі виходячи з наявних тренувальних даних та її архітектури).

2.6.6 Щільна нейронна мережа

Оскільки двустороння мережа довгої короткотривалої пам’ятті видає багатовимірний вектор у якості результату, а очікуваний від нейронної мережі у випадку даної магістерської роботи результат є відношенням або невідношенням до конкретного класу – багатовимірний результат ДКЧП-шару має бути

приведений до одновимірного результату у проміжку $[0, 1]$. Інакше кажучи, після обробки даних ДКЧП-шаром має відбутись прийняття остаточного рішення. Це виконується за допомогою щільного повнозв'язного прошарку нейронної мережі.

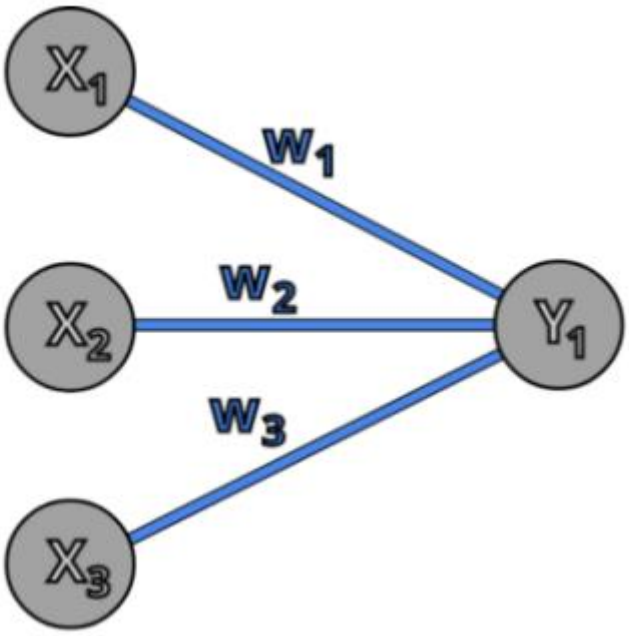

Рисунок 2.3 – Щільний прошарок нейронної мережі.

Щільний повнозв'язний прошарок є насправді стандартною класичною нейронною мережею, вихід якої розраховується за формулою:

$$y_i = H(W_{i-1}y_{i-1}),$$

де y_jє вихідним значенням нейрона, H є функцією активації, x_i є вхідним значенням нейрона попереднього прошарку, W_i є вагою нейрона попереднього прошарку.

2.7 Висновок до розділу

У цьому розділі магістерської роботи було описано математичні засади та алгоритми, використані або розроблені, описано відповідні специфіки, тощо.

Перша частина розділу описує та обгрунтовує розроблений алгоритм жадібного краулінгу, призначений для масового пошуку та збору новинних статей конкретного ресурсу.

У другій частині розділу розібрано алгоритм обробки та підготовки даних для застосування у машинному навчанні.

У третій частині було викладено та математично пояснено TF-IDF метрику тексту та показано псевдокод процесу трансформації тексту у розріджений TF-IDF вектор для споживання алгоритмами машинного навчання.

Четверта частина розділу пояснює алгоритм навчання токенізатора словнику на корпусі текстів та показує процес трансформації тексту у розріджений словарний вектор.

П’ята частина розділу викладає алгоритм навчання Пасивно-Агресивного Класифікатора та обгрунтовує його схожість та відмінність з алгоритмом опорних векторів.

Шоста частина розділу роз’яснює математичні принципи функціонування рекурентних нейронних мереж як загального класу та, зокрема, мереж Довгої Короткочасної Пам’ятті. Також було обгрунтовано необхідність застосування саме ДКЧП-мереж, пояснено відмінність двусторонніх ДКЧП-мереж від звичайних та важливість їх використання у обробці натуральних мов. У кінці частини було описано у тому числі математичні засади та необхідність застосування Ембедінгів та щільних мереж при роботі з рекурентними та ДКЧП-мережами.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СКРАПІНГУ ТА МАШИННОГО АНАЛІЗУ ОТРИМАНИХ ТЕКСТІВ

3.1 Екстрактор

Відповідно описаного в попередньому розділі алгоритму було реалізовано ресурс-агностичний скрапер на мові програмування JavaScript для платформи Node.js.

Вибір було зроблено серед двох популярних платформ імплементації скрапінг-утиліт: Python та JavaScript. Серед параметрів цих платформ було звернено особливу увагу на швидкодію у роботі з асинхронним мережевим вводом-виводом, необхідним для вчасного завантаження веб-сторінок та швидкодію у роботі з загальними обчислювальними операціями, необхідну для обробки та підготовки текстів.

Вважаючи набагато більш пропрацьовану та гнучку модель роботи з асинхронними операціями взагалі та зокрема асинхронними операціями вводу/виводу та загально кращу швидкодію як в умовах важкої роботи з вводом/виводом, так і кращу швидкодію в умовах роботи з великою кількістю обчислень [16][17] було обрано платформу Node.js та JavaScript.

Скрапер було реалізовано у вигляді консольного застосунка, який приймає на вхід домен, який необхідно краулити, та частки виключення, за якими відфільтровуються посилання всередині домену які потрібно проігнорувати. Більшу частину часу роботи скрапера займають завантаження сторінки та таймаут очікування, яке необхідне для уникнення блокування користувача скрапера ресурсом. Тому, на одній машині може бути запущено одночасно декілька скраперів, напрямлених на різні ресурси, при чому машина може залишатись працездатною для інших цілей через низькі потреби скраперів у обчислювальних потужностях.

Виходом скрапера є JSON-файл з зібраними статтями, посиланнями на них та їх заголовками, готовий для обробки подальшими програмними засобами.

3.2 Скрипт підготовки даних

Підготовку даних для навчання нейронних мереж, відповідно вищезазначеним швидкодійним причинам також було вирішено виконати на мові програмування JavaScript для платформи Node.js.

Скрипт підготовки даних є мапінг-функцією, яка приймає набір JSON-файлів з відповідними мітками, та віддає JSON-файл з обробленими текстами та розставленими мітками. За своєю структурою реалізація алгоритму є базовим маппером, який застосовує до текстів у масиві RegExp трансформації, та відфільтровує слова на факт існування у наборі шумових слів.

Як приклад роботи цього алгоритму можемо показати такий текст, помічений в даних як приклад неякісної журналістики та представлений на декількох сумнівних сайтах[2]:

«Потому что он выглядит, как бандит, говорит, как бандит и ведет себя, как бандит. Подельник Петра Порошенко, его старый армейский товарищ Игорь Кононенко — сегодня главный претендент на премию Дарвина в Украине. Колличество идиотских поступков, совершенных этим персонажем, стремительно переходит в качество. То самое качество, которое, например, дерьмо отличает от золота. В смысле — «молчание — золото», а словоблудие...»[3].

Після роботи цього алгоритму текст є позбавленим розмітки та шумових слів, та приведений до одноманітної форми:

«выглядит бандит бандит ведет бандит подельник петра порошенко старый армейский товарищ игорь кононенко главный претендент премию дарвина

[2] За оцінкою авторів роботи, керуючись довідкою[18]

[3] Важливо зауважити, що автори роботи не відповідають за інформацію, вказану в даних, використаних у дослідженні та не вважають коректними як деякі з мовних засобів, використані у вищезазначених даних, так і факт публікації та видачі такого роду матеріалів за журналістику. Форматування, орфографія, тощо збережені.

украине колличество идиотских поступков совершенных персонажем стремительно переходит качество самое качество которое например дерьмо отличает золота смысле молчание золото словоблудие»

3.3 Класифікатор на основі Пасивно-Агресивного класифікатора

Для підготовки даних для навчання пасивно-агресивного класифікатора було використано вищезазначений алгорим підготовки даних.

Реалізацію алгоритму трансформації даних для споживання класифікатором та сам класифікатор було вирішено запозичити з бібліотеки засобів машинного навчання Scikit-Learn, оскільки дана бібліотека є розповсюдженим інструментом роботи з підготовкою даних у машинному навчанні та роботи з класичними інструментами машинного навчання.

Для трансформації даних у векторну форму для тренування пасивно-агресивного класифікатора було обрано TF-IDF-векторизатор, оскільки очевидним є те, що пасивно-агресивний класифікатор не має можливостей для роботи з контекстами. TF-IDF-векторизатор є ідеальним вибором у таких випадках, оскільки дозволяє відслітковувати наявність та значимість у тексті слів-маркерів, які часто присутні у маніпулятивних та дезінформативних текстах.

```
(0, 294004)   0.04019485292599287
(0, 293964)   0.0209785796030312
(0, 293032)   0.08284744364924448
(0, 293020)   0.043721866882061165
(0, 293006)   0.031345028767366535
(0, 292518)   0.03483149864942345
(0, 292517)   0.039976628872608874
(0, 292490)   0.0632096625113808
(0, 292309)   0.20328328831465334
(0, 291511)   0.02624630635573961
(0, 286051)   0.0235302179671411
(0, 286027)   0.017548741861845945
(0, 284659)   0.1622109056464288
(0, 284640)   0.10999516969640209
(0, 281420)   0.03297041396344615
(0, 277323)   0.09208055133316215
(0, 274588)   0.03193221244749106
(0, 274329)   0.04413585861602509
(0, 274314)   0.05165306296475679
(0, 273936)   0.14370207669483673
(0, 273564)   0.04142372182462224
(0, 271738)   0.023461482964424078
(0, 270889)   0.05337929927639044
(0, 270552)   0.025646790922479255
(0, 270174)   0.035403407395228716
:     :
(2014, 15731) 0.013010107904457018
(2014, 15243) 0.04142203600076335
(2014, 15221) 0.05281765804806933
(2014, 15220) 0.021694788424202496
(2014, 15215) 0.020230538091164894
(2014, 14388) 0.011600353710359746
(2014, 14326) 0.01038842743813711
(2014, 14325) 0.009324578663618107
(2014, 13923) 0.023345386166730906
(2014, 13892) 0.01765903195140885
(2014, 13872) 0.019984602400195672
(2014, 13855) 0.028873167448012015
(2014, 13843) 0.01825091410352507
(2014, 12855) 0.01083550972284745
(2014, 12563) 0.024193910081873913
(2014, 12488) 0.04046107618232979
(2014, 12408) 0.022687219885508137
(2014, 12325) 0.01651722580685447
(2014, 12202) 0.012347509586918012
(2014, 11940) 0.04946334217919566
(2014, 11736) 0.02373923962829219
(2014, 11358) 0.011705329901002443
(2014, 11303) 0.02373923962829219
(2014, 9368)  0.02373923962829219
(2014, 4195)  0.021694788424202496
```

Рисунок 3.1 – Приклад виходу TF-IDF-векторизатора

На рисунку 3.1 відображена проекція розрідженої матриці, яка представляє вихід TF-IDF-векторизатора. Кожному стовпцю матриці відповідає один текст з набору вхідних даних, кожному рядку відповідає слово у словнику

векторизатора. Клітинками матриці є значення TF-IDF для конкретного слова в конкретному тексті відносно усього корпусу документів. Розрідженою матриця є через те, що не у всіх текстах наявні усі слова словника.

Імплементацію Пасивно-Агресивного класифікатора також було запозичено з бібліотеки засобів Scikit-Learn. Ця імплементація як для навчання, так і для реальної роботи сприймає розріджену матрицю, яка є виходом TF-IDF-векторизатора.

Навчання було проведено та результати протестовано за допомогою методу навчальної-тестувальної вибірок з часткою тестувальних даних у 20%.

Виходом класифікатора є одномірний вектор класу, який містить у собі значення 1 чи 0, яке характеризує клас новини.

3.4 Класифікатор на основі ДКЧП-мережі

Для реалізації класифікатора на основі нейронної мережі довгої короткочасної пам'яті було обрано фреймворк Keras з бекендом у Tensorflow, оскільки така комбінація є індустріальним стандартом в області розробки рішень, які включають машинне навчання та нейронні мережі, та мають відмінні характеристики з гнучкості та портативності. Зокрема, Tensorflow дозволяє збереження та конвертацію вагів та структури нейронної мережі для використання в контексті веб-сайтів та браузерних додатків.

Першим етапом підготовки тексту для споживання нейронною мережею є (після препроцесингу, описаного в розділі 3.2) є токенізація тексту, алгоритм якої описаний у розділі 2.4. Імплементацію цього алгоритму для навчання нейронної мережі було запозичено з бібліотек фреймворку Keras, але для перенесення нейронної мережі в браузерний додаток алгоритм токенізації було відтворено на мові програмування JavaScript.

```
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0 1349  694 1716
 331    3    2  383  456    3    2    3    2   43  273 1295  132  607
 754    3    2  470   14  593   55 1093  754  626  710  789 1349  380
 839  966  441  686 1366  104   84  380  607   16    2 1404   43   55
 541  981  429  710   18  186  273  586  271 1052 1404   43  272 1835
 573  481  969  606 1091 1907   43  960 1739 1728  328  553 1618   24
 831 1349  382   25   74   43  502   48  160   55  287 1245  742  408
 369  386  789  963  310 1446  906 1730  762  972 1906  413  579  460
 643   18 1209  331 1000 1296 1349  694 1716  331    3    2  383  456
   3    2    3    2   43  273 1295  132  607  754    3    2  470   14
 593   55 1093  754  626  710  789 1349  380  839  966  441  686 1366
 104   84  380  607   16    2 1404   43   55  541  981  429  710   18
 186  273  586  271 1052 1404   43  272 1835  573  481  969  606 1091
1907   43  960 1739 1728  328  553 1618   24  831 1349  382   25   74
  43  502   48  160   55  287 1245  742  408  369  386  789  963  310
1446  906]
```

Рисунок 3.2 – Частина вихідного вектору токенізатора

Виходом токенізатора є розріджений вектор індексів слів у словнику. Розрідженим вектор є через те, що слова, які не входять в словник заміщаються 0.

Другим етапом підготовки тексту є призведення до сталого розміру, який був обчислений відповідно середнього розміру тексту у датасеті. Результуючою

розмірністю вхідних текстів було таким чином визначено 4678. Тексти, які перевищують такий розмір у словах обрізається до відповідності цій розмірності. Тексти, які є меншими за цю розмірність доповнюються нулями в початок вектору до досягнення необхідної розмірності.

Токенізовані та приведені до одноманітного розміру тексти направляються в нейронну мережу.

Першим прошарком нейронної мережі є Ембедінг-шар. Ембедінг-шар, як зазначено у розділі 2.6.5, будує словник трансляції окремих слів в їх вектори. Ці вектори є об'єктом тренування нейронної мережі, і дозволяють їй вивчити близькі за використанням одне до одного слова.

Входом ембедінг-шару є матриця, яка складається з розріджених векторів токенізованого тексту. Розмірність вхідної матриці, відповідно, становить $(n, 4678)$, де n є кількістю текстів, які підлягають класифікації в даний момент часу.

Виходом ембедінг-шару є тривимірна щільна матриця, у якій кожному слову кожного вхідного тексту відповідає 16-мірний вектор. Відповідно, розмірність вихідної матриці ембедінг-шару становить $(n, 4678, 16)$, де n є кількістю текстів, які підлягають класифікації в даний момент часу.

Другим прошарком нейронної мережі є слой нейронів двусторонньої довгої короткочасної пам'ятті, математичні засади якої описано в розділі 2.6.

Даний прошарок обробляє вхідну гіперматрицю $(n, 4678, 16)$ як часову послідовність слів у два паралельних проходи – з початку в кінець та з кінця в початок для повноцінної обробки попередніх та майбутніх контекстів слова.

Розміром ДКЧП-прошарку було обрано 64 блоки.

Для формулювання кінцевого результату роботи мережі з 64-мірного вектору-результату ДКЧП-прошарку було застосовано стандартний повнозв'язний прошарок у 32 нейрони з ReLU активацією для формулювання рішення системи.

Виходом нейронної системи є один нейрон повнозв'язного щільного типу з сигмоїд-активацією для отримання чіткого результату у діапазоні $(0, 1)$.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 4678, 16)          32000
_________________________________________________________________
dropout_1 (Dropout)          (None, 4678, 16)          0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)                41472
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 75,585
Trainable params: 75,585
Non-trainable params: 0
_________________________________________________________________
None
```

Рисунок 3.3 – Кінцева структура мережі, вихідні параметри прошарків та кількості тренувальних параметрів.

Для уникнення випадків перепристосування моделі було додано викидні прошарки нейронної мережі. Викидні прошарки мережі функціонують тільки під час тренування, скидаючи випадкові входи з частотою $rate$ наступних прошарків у 0, та масштабуючи інші на визначений коефіцієнт $\frac{1}{1-rate}$.

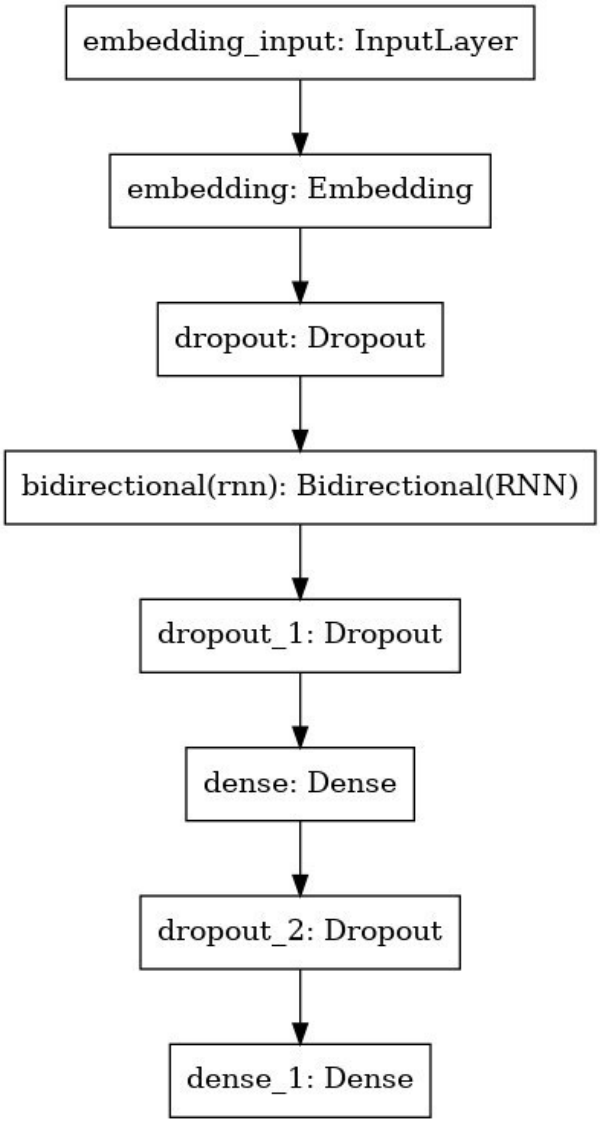

Рисунок 3.4 – Спрощений вигляд графу отриманої мережі.

3.5 Альтернативний скрапер

Для оцінки результативності роботи розробленого скрапера відносно існуючих в індустрії скрапінг-застосунків було обрано скрапер ScrapingHub. ScrapingHub є провайдером засобів для краулінгу та іншого видобутку даних у мережі інтернет, та надає API для запуску власних скраперів та краулерів на їх хмарному рішенні, побудованому таким чином аби уникнути блокування видобуваючих програм сайтами-цілями.

Помітним мінусом ScrapingHub є необхідність імплементації власних краулерів, які б працювали на їх хмарному рішенні або за допомогою їх скрапінг-проксі.

Головним продуктом ScrapingHub є скрапігн-API, яке дозволяє автоматично видобувати дані з веб-сторінок, вказавши лише тип необхідних даних та посилання на сторінку, з якої дані мають бути видобуті. Чистота даних досягається за допомогою багаторівневого алгоритму з використанням очистки алгоритмами машинного навчання.

У складі ScrapingHub Automatic Data Extraction API наявні такі скрапінг-цілі:

- News API – автоматично видобуває статті та новини за цільовими посиланнями
- Product data API – дозволяє видобувати інформацію про товар, наявну на переданій сторінці
- Job postings API – інформація про наявні вакансії та резюме людей, які шукають роботу.
- Vehicle data API – інформація про автомобілі, виставлені до продажу на сторінці.
- Product review API – збір оглядів та відгуків на товар у конкретному магазині
- Інші: Hotel, Flight Information, Real Estate, Restaurant, Reviews, Sellers, Events, Locations/Places, Medicine, Educational courses, тощо.

Скрапінгові та краулінгові послуги ScrapingHub відрізняються високою якістю та чистотою даних, які забезпечуються багатоступеневим процесом забезпечення якості, який складається з багатьох автоматичних та автоматизованих кроків, та у деяких випадках включає в себе також і ручний.

Рішення ScrapingHub в першу чергу розраховані на використання великими корпоративними клієнтами, які вимагають витягнення даних у об’ємах від 100 тисяч запитів в місяць, та у випадках особливо великих клієнтів

включають в себе винаймання замовником цілої команди розробників для розробки та підтримки спеціалізованого рішення, функціонал та властивості якого будуть підходити саме під потреби конкретного клієнта та інтегруватись з його інформаційними системами.

3.6 Порівняння результатів.

За допомогою скрапера-краулера, розробленого у процесі виконання даної магістерської роботи було зібрано 39482 статей загалом, з яких 16129 належать до ресурсів з підтвердженою репутацією, та 23353 належать до ресурсів які не відповідають стандартам журналістики відносно критеріїв [18].

Альтернативний скрапер не має вбудованого краулінг-функціоналу, і, відповідно, мав бути інтегрованим з або краулером, який надається ScrapingHub, або з third-party краулером. Краулер, який надається ScrapingHub є окремою послугою, яка не поставляється у комплекті з скрапером та має бути інтегрований вручну користувачем API, що є складним процесом який потребує розробки повноцінного програмного засобу. Також, на краулер ScrapingHub не надається легкодоступний випробувальний період.

Виходячи з описаних в попередньому параграфі причин було прийнято рішення використати зібрані розробленим нами краулером посилання та зняти дані за ними. Таким чином, використовуючи випробувальний період скрапера ScrapingHub було зібрано 8360 статей, з яких 5702 належать до ЗМІ з репутацією та 2658 належать маніпулятивним ЗМІ. Таким чином, було сформовано датасет аналогічних зібраним розробленим в цій магістерській роботі скрапером статей.

Було помічено, що скрапер від ScrapingHub видає стабільно помітно чистіші дані. Оскільки розроблений у цій магістерській роботі скрапер є простим та суто аналітичним алгоритмом, який не містить інтелектуальної очистки даних у результати роботи скрапера можуть потрапляти зайвий текст, який відноситься не до матеріалу, а до загального функціонування ресурсу. Ці помилкові дані для

подальшого навчання нейронної мережі було видалено в напівавтоматичному режимі. Приклади помилкових даних наведено в таблиці 3.1.

Таблиця 3.1 – Помилкові дані у роботі розробленого скрапера

Ресурс	Результат роботи розробленого скрапера	Результат роботи скрапера ScrapingHub
https://40ka.info/?p=13004	Верховний Суд визнав, що президент України зобов’язаний спілкуватися державною мовою під час виконання службових обов’язків, але вирішив не карати Володимира Зеленського за використання російської … вимоги не підлягають розгляду в порядку адміністративного судочинства. Комментарии закрыты.	Верховний Суд визнав, що президент України зобов’язаний спілкуватися державною мовою під час виконання службових обов’язків, але вирішив не карати Володимира Зеленського за використання російської … вимоги не підлягають розгляду в порядку адміністративного судочинства.

Продовження таблиці 3.1

https://ua-news.liga.net/world/news/e-13-dniv-schob-vikonati-tri-umovi-tihanovska-visunula-lukashenku-ultimatum	Світлана Тихановська дала невизнаному президенту Білорусі 13 днів на виконання трьох вимог опозиції і попередила про національний страйк\n\nЛідерка білоруської опозиції Світлана Тихановська висунула ультиматум самоправозглашенному президенту Білорусі Олександру Лукашенку. Про це вона повідомила у себе в Telegram. … Тихановська говорила – \"режим зрозумів, що з білорусами потрібно розмовляти, тому Лукашенко поїхав в СІЗО КДБ\	Світлана Тихановська дала невизнаному президенту Білорусі 13 днів на виконання трьох вимог опозиції і попередила про національний страйк Лідерка білоруської опозиції Світлана Тихановська висунула ультиматум самоправозглашенному президенту Білорусі Олександру Лукашенку. Про це вона повідомила у себе в Telegram. … Тихановська говорила – \"режим зрозумів, що з білорусами потрібно розмовляти, тому Лукашенко поїхав в СІЗО КДБ\". Адрес страницы с ошибкой: Текст с ошибкой: Ваш комментарий или корректная версия:

На зібраних датасетах було натреновано описані в розділах 2.5, 2.6, 3.3, 3.4 алгоритми, засновані на Пасивно-Агресивному Класифікатори та нейронних мережах Довгої Короткочасної Пам’ятті.

На датасеті, зібраному за допомогою скрапера ScrapingHub було досягнено таких результатів:

- Пасивно-агресивний класифікатор: 0.975
- ДКЧП-мережа: 0.958

На датасеті, зібраному за допомогою розробленого у процесі виконання даної магістерської роботи було досягнено таких результатів:

- Пасивно-агресивний класифікатор: 0.979
- ДКЧП-мережа: 0.966

Виходячи з описаних вище результатів та порівняльного аналізу, маємо повне право заключити що описаний у розділах 2.1, 3.1 краулер-скрапер є повноцінним програмним засобом, який в контексті пошуку та видобування новин для тренування алгоритмів машинного навчання показує результати, співставні з такими у існуючих в індустрії рішень. Варто зауважити що особливістю розроблених алгоритму та програмного засобу є суміщені процеси краулінгу та скрапінгу.

3.7 Опис функціональності побудованого браузерного додатку

Використовуючи натреновану ДКЧП-мережу (описану в 2.6 та 2.4), а також спрощених варіантів алгоритмів, описаних у 2.1 та 2.2 було створено додаток для браузера Chrome. Після інсталяції даний додаток перевіряє кожну сторінку, на яку заходить користувач системи на те, чи є відкрита сторінка статтею українською чи російською мовами, та якщо на сторінці було знайдено статтю – обробляє її та класифікує за допомогою вбудованої в додаток нейронної мережі, після чого виводить один з трьох банерів, які нагадують користувачеві про обережне споживання новинної інформації.

Банери зображено на рисунках 3.5, 3.6, 3.7.

Рисунок 3.5 – Попередження про вірогідно неякісний матеріал[4]

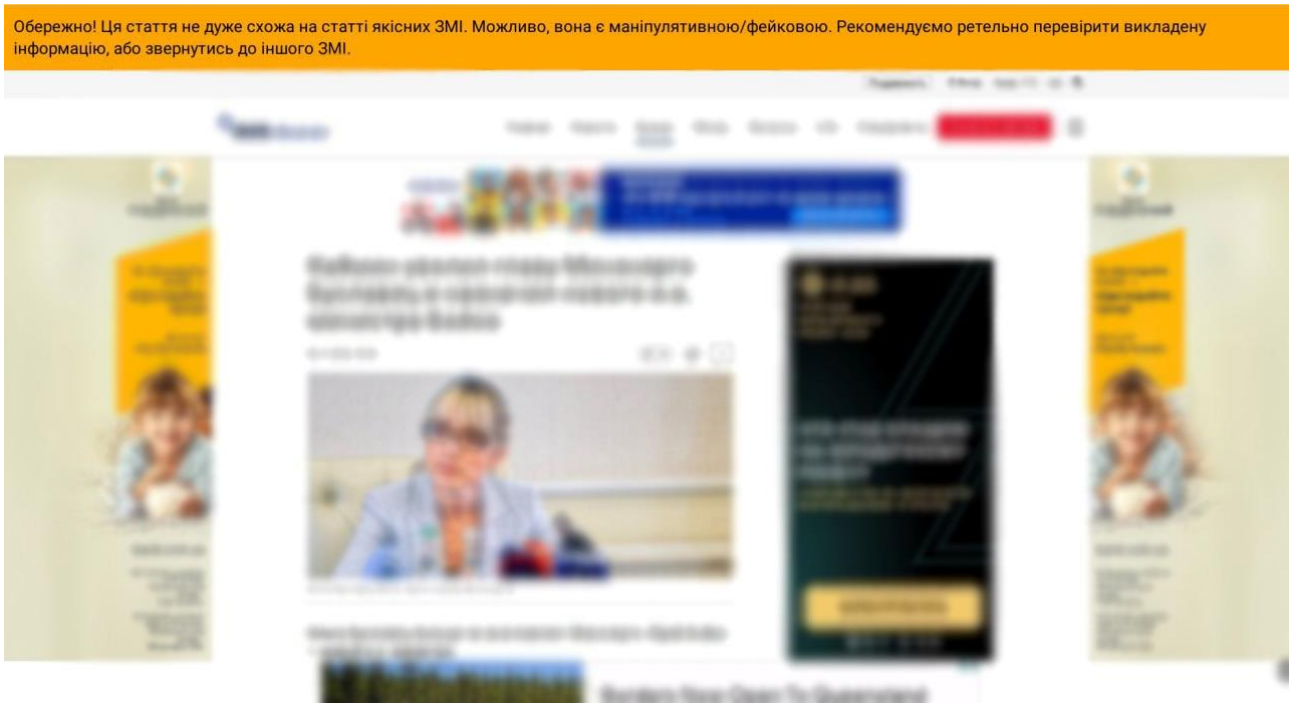

Рисунок 3.6 – Попередження про імовірно неякісний матеріал

[4] Тут і надалі конкретні матеріали було розмито у графічному редакторі. Додаток не накладає цей ефект самостійно.

Рисунок 3.7 – Попередження про імовірно якісний матеріал

3.8 Опис архітектури побудованого браузерного додатку

Розроблений браузерний додаток складається з двох принципових частин:

- Контент-сценарій
- Фоновий сценарій.

Дані сценарії спілкуються між собою за допомогою міжпроцесових повідомлень середовища виконання Google Chrome.

Структура повідомлення до фонового сценарію:

```
{
    action: 'TF_ESTIMATE_ARTICLE',
    article: 'string'
}
```

Відповіддю фонового сценарію на даний запит є число у межах $(0, 1)$, яке визначає належність відправленої статті до схожих на правдиві чи на маніпулятивні.

3.8.1 Контент-сценарій

Контент-сценарій підключається до кожної завантажуваної браузером сторінки, та проводить пошук та видобуток статті з завантаженої сторінки спрощеним варіантом алгоритму, описаного у розділі 2.1. У випадку наявності статті на сторінці стаття відправляється до фонового скрипта для проведення її класифікації. Відповідно результатів класифікації контент-скрипт відображає на сторінці баннер, який попереджає користувача про потенційну неякісність відображеної новини. Структуру контент-сценарію описано в таблиці 3.2.

Таблиця 3.2 – Опис функцій контент-скрипта

Назва функції	**Опис**
findArticle()	Виконує пошук та видобування статті з поточної веб-сторінки
getArticleHeader()	Шукає та виконує початкове форматування заголовку статті
getArticleContent()	Шукає на поточній веб-сторінці зміст статті, та очищує його від зайвих елементів (посилання, кнопки, скрипти, стилі, картинки, тощо
onPageLoad()	За наявності на сторінці статті відправляє її на аналіз у фоновий скипт додатку
onAnalysisReady()	Отримує результат класифікації від фонового скрипта та відображає на сторінці банер, відповідний отриманому класу

3.8.2 Фоновий сценарій

Фоновий сценарій виконує завантаження списку стоп-слів та нейронної мережі з файлів, які постачаються в пакеті браузерного додатку та використовує їх для класифікаціїї наданих контент-сценарієм статей.

Класи та функції, наявні у фоновому сценарії описано у таблицях 3.3 та 3.4.

Таблиця 3.3 – Опис класів фонового скрипта додатку

Клас	**Опис**
Tokenizer	Виконує очищення та токенізацію вхідного тексту.
FakeClassifier	Виконує завантаження вагів та даних, необхідних для функціонування токенізатора. Виконує класифікацію наданого контент-скриптом тексту за допомогою завантаженої передтренованої ДКЧП-мережі.

Таблиця 3.4 – Опис методів класів та глобальних функцій фонового скрипта додатку

Клас	**Метод**	**Опис**
Tokenizer	constructor	Виконує початкову ініціалізацію властивостей класу
Tokenizer	cleanText(text)	Виконує очистку тексту від зайвих пробілів, знаків розмітки, приводить до нижнього регістру та очищує від стоп-слів.

Продовження таблиці 3.4

Tokenizer	textToSequence(text)	Виконує підготовку тексту за допомогою власного методу cleanText та формує вхідний вектор нейронної мережі за допомогою наявного в класі трансформаційного словника
Глобальний контекст	tokenizerFromJson(json, stopwords)	Створює новий інстанс класу Tokenizer та наповнює його трансформаційний словник та словник стоп-слів, та повертає готовий для роботи інстанс класу.
Глобальний контекст	padSequence(sequence, maxLength)	Виконує вирівнювання вхідного вектора нейронної мережі. У випадку перевищення текстом вхідної розмірності мережі обрізає його до необхідної довжини, інакше – доповнює нулями в початок до досягнення потрібної розмірності

Продовження таблиці 3.4

FakeClassifier	constructor()	Ініціює завантаження моделі та інших необхідних для функціонування даних
FakeClassifier	loadModel()	Виконує завантаження вагів та структури класифікаційної моделі, словників необхідних для функціонування токенізатора. Створює інстанс токенізатора для подальшого використання класом.
FakeClassifier()	predict()	Виконує токенізацію та очистку тексту, перетворює його в тензор-об'єкт та виконує класифікацію за допомогою завантаженої моделі.
Глобальний контекст	onMessage()	У випадку отримання правилього повідомлення виконує класифікацію отриманого тексту за допомогою інстансу класу FakeClassifier та відправляє відповідь у вигляді результату класифікації.

3.9 Інструкція з установки браузерного додатку

Для встановлення даного додатку з файлу розширення .crx користувач має перейти на сторінку `chrome://extensions` та увімкнути режим розробника за допомогою перемикача у правому верхньому кутку (Рисунок 3.8).

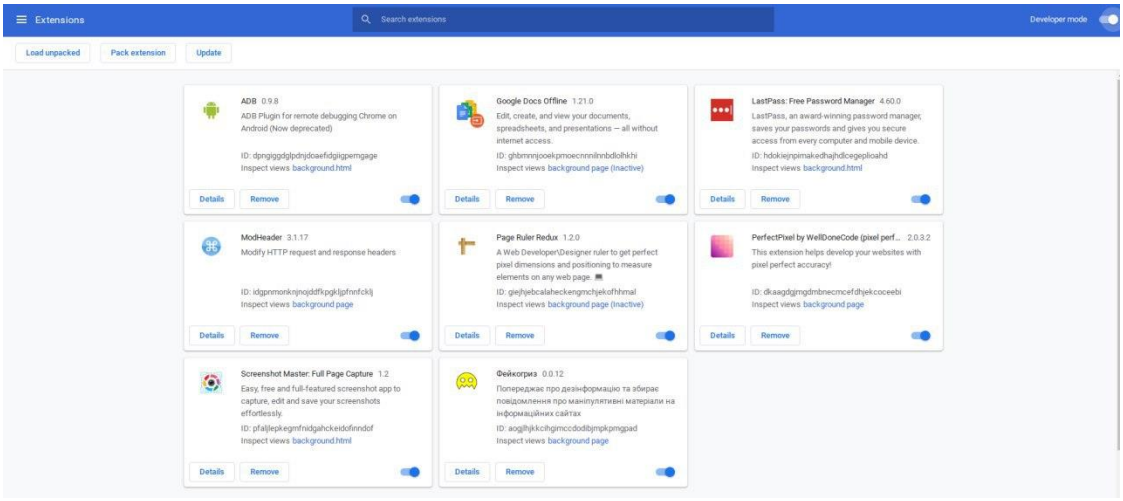

Рисунок 3.8 – Сторінка розширень браузера Google Chrome

Увімкнувши режим розробника користувач має оновити сторінку та перетягнути файл додатку на сторінку до появи відповідної підсвітки, яка відображена на Рисунку 3.9.

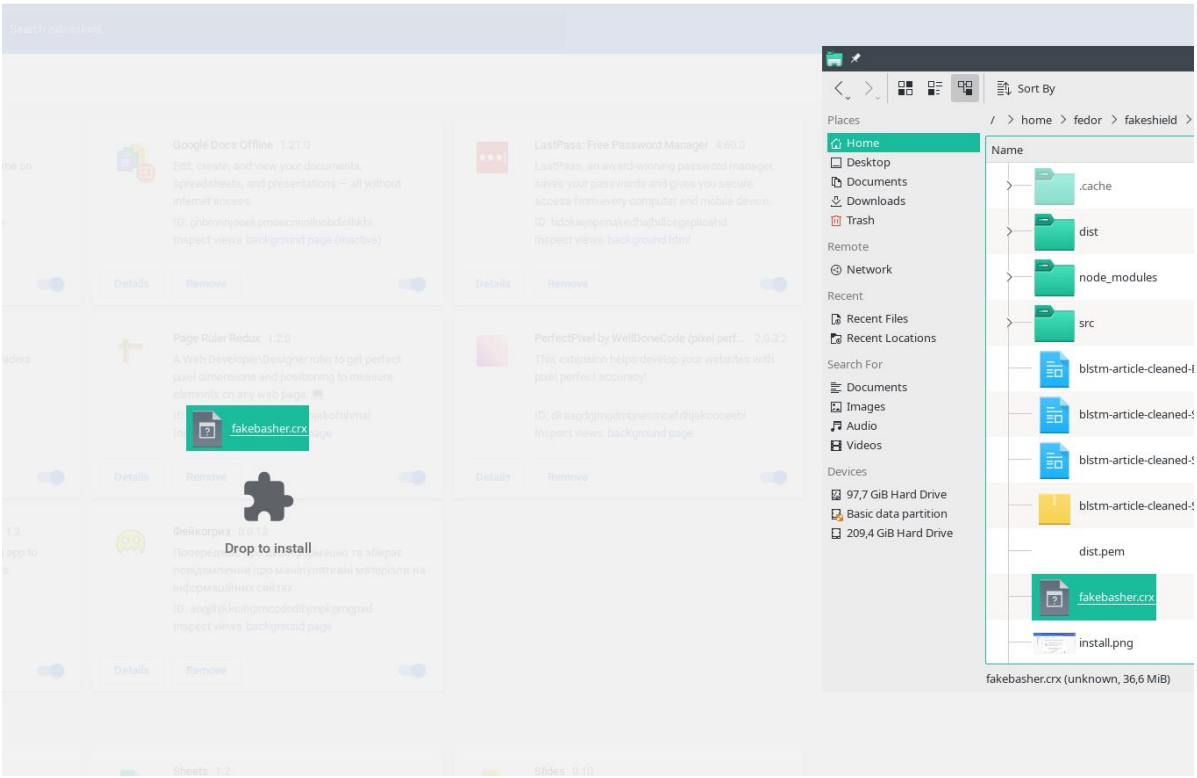

Рисунок 3.9 – drag-and-drop підсвітка установки браузерного додатку

Після перетягнення додатку на сторінку-список додатків браузером буде відображено діалогове вікно встановлення додатку, на якому користувач має натиснути «Продовжити» для остаточного встановлення додатку (Рисунок 3.9).

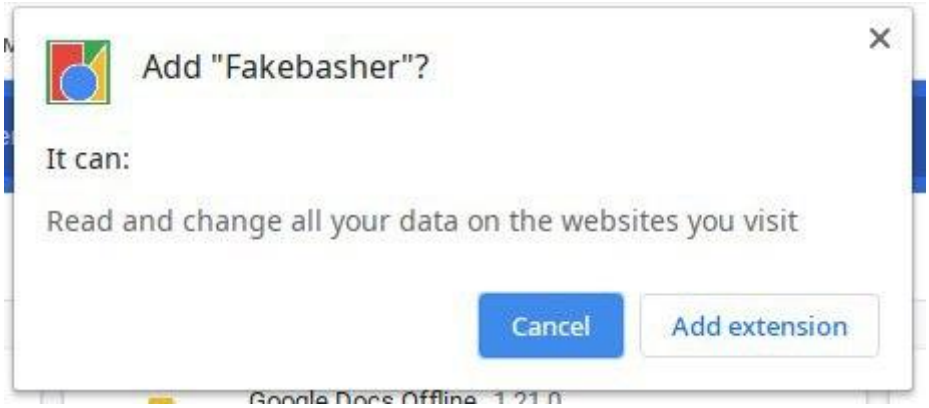

Рисунок 3.9 – діалогове вікно встановлення додатку

Після натискання кнопки користувач може почати використання додатку, відкриваючі новинні ресурси.

3.10 Інструкція з використання скрапера

Для початку роботи з скрапером користувач має видобути пакет програмного забезпечення та виконати встановлення залежностей за допомогою команди `npm install`.

Після встановлення залежностей скрапер-краулер є готовим до використання.

За допомогою команди `--help` користувач може вивести довідку про користування пакетом (Рисунок 3.10). Ця довідка містить список параметрів скрапера-краулера.

```
[fedor@flhdg fakeshield]$ ./scraper.js --help

NewsScraper

  A greedy scraper-crawler for news websites

Options

  -d, --domain link to a website root   The website to crawl-scrape.
  -e, --exclude string                  The strings that if found in a link will ignore the link
  -o, --output file                     The file to write the results to
  -s, --sleep ms                        How much to wait between requests
  -u, --useragent string                Useragent to check against when looking up robots.txt. Defaults to Googlebot
  --help string                         Print this usage guide.
```

Рисунок 3.10 – Довідка про користування скрапером-краулером

Головним та обов'язковим параметром скрапера є `--domain`. У цей параметр користувач має передати посилання на веб-ресурс, дані з якого потрібно зібрати. Також, цей параметр є стандартним, тобто його можно застосовувати без зазначення імені, тому:

```
./scraper.js --domain http://example.com
./scraper.js -d http://example.com
./scraper.js http://example.com
```

Є еквівалентними виразами використання інструменту.

Другим параметром скрапера є `--exclude`. Цей параметр задає набір фільтрів для посилань, які краулер має проігнорувати на доповнення до посилань на інші сайти та посилань, які задані до ігнорування у robots.txt цільового ресусу. Наприклад:

```
./scraper.js https://example.com --exclude ex1 --exclude /another/url
```

Така команда збиратиме дані з усіх сторінок веб-ресурсу, які дозволені до збирання файлом robots.txt ресурсу, та не включають в себе строки ex1 та /another/url

Третім параметром скраперу є `--output`. Даний параметр задає назву файлу, до якого буде записано дані, витягнені з сторінок опрацьованих скрапером, наприклад:

```
./scraper.js https://example.com --output o.json
```

Така команда зберігатиме видобуті дані до файлу o.json.

Четвертим параметром скраперу є `--sleep`. Цей параметер визначає проміжок часу у мілісекундах, який скрапер має вичекати перед переходом на наступну сторінку. Даний параметр задається для уникнення блокування адреси та пристрою, з якого виконується краулінг, оскільки занадто часті запити привертають увагу систем захисту від атак виду DDOS.

Останнім параметром скраперу є `--useragent`. Даний параметр задає значення UserAgent для запитів на цільовий ресурс та для визначення дозволених та заборонених посилань відносно файлу robots.txt ресурсу.

Після запуску скрапера буде відбуватись цикл жадібного краулінгу-скрапінгу відносно алгоритму, описаного в 1. На кожному кроці циклу програма виводитиме поточний стан: поточне посилання, яке обробляється скрапером-краулером, та поточну кількість доступних до обробки посилань, відкритих скрапером-краулером (Рисунок 3.11). Також на кожному кроці циклу програма зберігатиме результати своєї роботи до файлу output.json, або вказаного у параметрі `--output`.

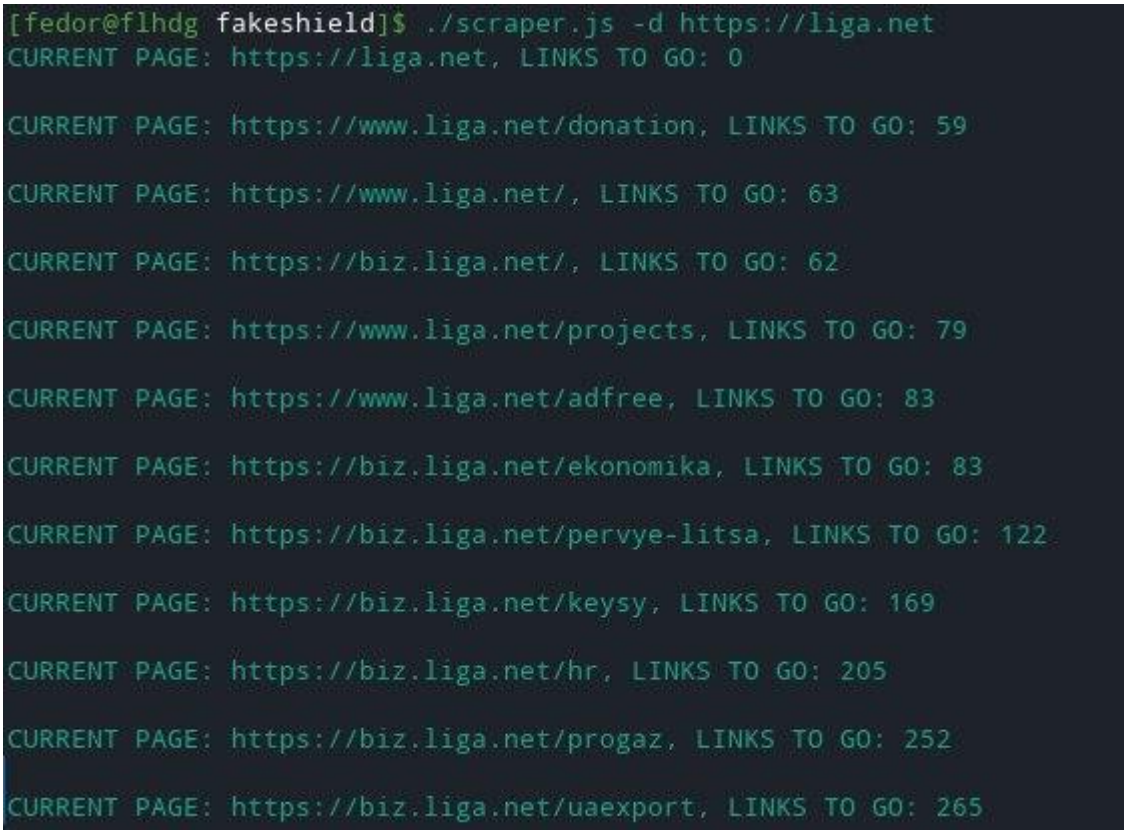

Рисунок 3.11 – Робота скрапера-краулера його вивід.

3.11 Інші застосування

Скрапер-краулер, описаний в розділах 2.1 та 3.1 може застосовуватись для збору та аналізу новинних даних з цілями, відмінними від цілей даного проекту.

Класи, описані в розділі 3.8 мають відкритий та зрозумілий інтерфейс, який дозволяє передавати на аналіз довільні тексти. Таким чином, розроблений програмний код може бути також імпортований в інші програмні застосунки та веб-сервіси, наприклад, форуми, чат-сервіси, тощо для допомоги у автоматичній фільтрації контенту, створеного користувачами вищеописаних програмних засобів, аналізу сторінок, посилання на які поширюють користувачі, тощо.

3.12 Висновки до розділу

У цьому розділі магістерської роботи було описано реалізації описаних у розділі 2, зазначено досягнені результати розробки та надано інструкції з користування результами розробки.

Частини 1-4 описують реалізації та деталі реалізації алгоритмів та процесів, описаних у розділі 2 та показано деталі функціонування розроблених процедур.

Частина 5 розділу описує альтернативний скрапер, який було розроблено та застосовано для порівняння з описаним у розділі 2.1 та реалізованим у розділі 3.1 скрапером.

У частині 6 було порівняно результати застосованих у цій магістерській роботі скраперів та визначено, що описаний у розділі 2.1 скрапер-краулер є конкурентно спроможнім алгоритмом, та результати його застосування до тренування алгоритмів машинного навчання є співставними з такими у промислового скрапера, а обрані алгоритми класифікації є достатніми для досягнення поставленої задачі.

У кінці розділу було наведено описи функціональності та архітектури побудованих засобів та наведено інструкції користувачів. Також було наведено альтернативні варіанти застосування частин побудованих програмних засобів та описано наукову новизну отриманих результатів.

4 СТАРТАП-ПРОЕКТ

4.1 Опис основної ідеї проекту

У даному розділі проаналізовано та подано зміст ідеї стартап-проекту, розібрано можливі напрямки застосування, вигоди користувача та порівняно розроблений програмний засіб з існуючими аналогами.

Зміст ідеї – програмні засоби для допомоги у аналізі новинних матеріалів та попередження про неякісні новинні матеріали.

Таблиця 4.1 – Опис ідеї стартап-проекту

Зміст ідеї стартап-проекту	Напрямки застосування	Вигоди для користувача
Розпізнавання текстових матеріалів, схожих на неякісні новини.	Браузерний додаток	Попереджує користувача про сумнівну якість відкритого ним матеріалу
	Модуль для використання в інших програмних продуктах	Є щаблем у автоматичній модераційній або фільтраційній підсистемах новинного агрегатора, чат-сервісу, форуму, тощо.

Для порівняння пропонованого стартап-проекту з пропозиціями конкуруючих засобів було виконано такі кроки:

- Визначено техніко-економічні властивості пропонованого стартап-проекту

- Визначено приблизне коло конкурентів на ринку та визначено техніко-економічні властивості продуктів, які пропонуються
- Виконано порівняльний аналіз видобутих даних та відсортовано відносно сильних, нейтральних та слабких сторін

Результати виконаної у цьому напрямі роботи подано у таблиці 4.2

Таблиця 4.2 – Порівняння характеристик з конкурентними продуктами

	Характеристики		
	Пропонований проект	Фейкогриз	Trusted times
Сильна сторона пропонованого проекту	Робота з будь-якими матеріалами українською та російською мовами	Попереджує тільки про сайти, які має в власній базі даних, та не звертає уваги на конкретні матеріали	Працює тільки з матеріалами англійською мовою
Нейтральна сторона пропонованого проекту	Надає лише попереджувальну оцінку щодо відкритого контенту	Надає лише попереджувальну оцінку щодо відкритого контенту	Надає глибокий аналіз тексту включно з аналізом упередженості до персон, згаданих у матеріалі

Продовження таблиці 4.2

Слабка сторона пропонованого проекту	Є обчислювально важким, оскільки засновується на використанні нейронної мережі	Працює моментально, оскільки працює звіряючи домен з власною базою новинних видань	Працює достатньо швидко, оскільки процес аналізу відбувається на бекенді додатку.

4.2 Технологічний аудит ідеї проекту

В даному розділі аудитовано технології, від яких залежить реалізація ідеї проекту з надання оцінки якості новинних матеріалів.

Проаналізовано такі складові:

- Яка технологія є базовою у продукті?
- Чи доступна ця технологія, чи вона вимагає її винайдення чи покращення?
- Чи легко ці технології доступні?

Результати такого аналізу у застосуванні до пропонованого проекту наведено у таблиці 4.3.

Таблиця 4.3 – Технологічна здійсненність

	Ідея пропонованого проекту	Технологія, від якої залежить реалізація	Наявність технології	Доступність технології
1	Розпізнавання текстових матеріалів, схожих на неякісні новини.	Набір очищених та розмічених новинних матеріалів	Наявна, але потребує доповнення та більш ретельної очистки	Доступна
2		Засоби для розробки та інтеграції нейронних мереж	Наявна	Доступна
3		ДКЧП-мережі	Наявна	Доступна

З отриманих результатів технологічного аудиту та аналізу конкурентів пропонованого проекту можемо зробити висновок, що реалізація проекту з технічної точки зору реальна на базі існуючих та доступних технологій.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Для коректного планування напрямків розвитку проекту з урахуванням поведінки ринку, пропозицій конкуруючих гравців та запитів активних та потенційних споживачів необхідно у тому числі визначити ринкові можливості та ринкові загрози. Подальший аналіз буде стосуватись виключно українського ринку.

Першим етапом у такому аналізі є вияснення об’ємів попиту на пропонований продукт та динаміки ринку. Такий аналіз наведено в таблиці 4.4

Таблиця 4.4 Попередня характеристика потенційного ринку стартап-проекту

	Показник ринку	Значення показника ринку
1	Кількість головних гравців	1
2	Загальний обсяг продаж	0 (конкурент є безкоштовним і працює на пожертви та гранти)
3	Динаміка ринку	Відсутня (ринок ще не є сформованим)
4	Обмеження для виходу на ринок	Відсутня
5	Вимоги до сертифікації або стандартизації	Відсутні

Враховуючи пустоту ринку можемо зробити висновок, що хоч фінансово ринок і є невизначеним і невідомим, він все ж є достатньо привабливим для входження.

Наступним необхідно визначити категорії потенційних клієнтів та означити характерні їм риси і їх вимоги до продукту (таблиця 4.5).

Таблиця 4.5 – Категорії потенційних клієнтів пропонованого проекту

	Потреба	Аудиторія	Відмінності	Вимоги
1	Захист малообізнаних родичів/тощо від впливу неякісних новин	Технічно обізнана частина населення	Не готові платити відразу	Простота, доступність та помітність інтерфейсу. Стабільність роботи. Наявність безкоштовного періоду, або безкоштовної версії.
2	Фільтрація небажаного контенту	Розробники продуктів, які включають в себе відображення довільного контенту та контенту, згенерованого користувачами	Схильні до заключення тривалих домовленостей та мають необхідність у технологічній підтримці, готові платити за допрацювання продукту під себе	Наявність точної та доступної документації. Надійність роботи, швидкодія. Інтеграція системи з існуючим програмним забезпеченням

Наступним кроком аналізу ринку є підведення загрожуючих та сприяючих факторів поведінки ринку відносно пропонованого проекту. Результати наявні в таблиці 4.6

Таблиця 4.6 – Ринкові фактори

	Фактор	Вид	Зміст	Пропонована
1	Низький попит, неготовність платити	Загроза	Продукт не є життєво необхідним	Проведення рекламної та інформаційної кампаній про загрозу неякісних новинних матеріалів
2	Популярність системи серед (наприклад) журналістів	Можливість	Продукт підходить не тільки прямому споживачеві	Доповнення функціоналу для покращення відповідності системи до вимог нових категорій споживачів
3	Застосування маніпулятивними ЗМІ методик обману додатку	Загроза	Продукт не виконує поставлених задач	Створення відділку моніторингу для швидкого реагування та підлаштування системи для захисту від методик обману
4	Відсутність конкурентно спроможніх альтернатив	Можливість	Технологічна перевага проекту над конкурентами	Вихід на сусідні ринки

Також важливим є знання пропозицій та конкуренції на ринку. Такий аналіз має бути виконаний через призму існуючих видів конкуренцій, та

визначити у тому числі шлях просування подукту. Аналіз конкуренції наведено в таблицях 4.7 та 4.8.

Таблиця 4.7 – Аналіз конкуренції

Особливості конкуренції	Прояв особливості	Вплив на пропонований продукт
Конкуренція – монополістична, чиста монополія, олігополія	чиста	Просування продукту, укладання договорів пакетних підписок
Рівень конкурентної боротьби – локальний, національний, глобальний	локальний	Просування продукту по всій країні
Галузь – внутрішньогалузева, міжгалузева	Внутрішньогалузева	Покращувати точність, зручність, тощо
За товарами – між бажаннями, товарно родова, товарно-видова	Товарно-видова	Покращувати точність, зручність, тощо
Конкурентна перевага – цінова, нецінова	нецінова	Додавати метрики тексту, які відсутні у конкурентів
Інтенсивність – марочна, немарочна	марочна	Додавання загального функціоналу, недоступного конкурентам

Таблиця 4.8 – Галузева конкуренція, проаналізована за М. Портером.

Складові аналізу	Прямі конкуренти у галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Фейкогриз	TrustedTimes	Концентрація постачальників	Контроль якості	Ціна
Висновки	Визначити ступінь конкурентної боротьби з боку прямих конкурентів	Може вийти на ринок, якщо пристосує своє рішення до двомовного контексту України	Постачальники підлаштовуються під ринок	Клієнти обирають якісніші класифікатори	Обмеження для роботи через товари замінники

На основі аналізу конкуренції, характеристик ідеї проекту, вимог споживачів, ринкового середовища можемо визначити та обгрунтувати перелік факторів конкурентоспроможності.

Таблиця 4.9 – Обгрунтування факторів конкурентоспроможності

	Фактори конкурентоспроможності	Обгрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Динамічність обробки	Існуючі прямі конкуренти не мають алгоритмів, які працюють в режимі реального часу на будь-якому новинному ресурсі, який відкриває користувач
2	Робота з українським двомовним контекстом	Існуючі непрямі конкуренті не мають алгоритмів, які працюють з українською або російською мовами

За визначеними факторами конкурентоспроможності проводимо аналіз сильних та слабких сторін даного стартап-проекту (таблиця 4.10).

Таблиця 4.10 – Порівняльний аналіз слабких та сильних сторін

	Фактори конкурентноспроможності	Бали	Рейтинг продукту конкурентів у порівнянні з пропонованим продуктом						
			-3	-2	-1	0	+1	+2	+3
1	Динамічність обробки			+					
2	Робота з українським контекстом					+			
3	Швидкість обробки						+		

Фінальним кроком ринкового аналізу можливостей впровадження проекту є SWOT-аналіз, який представляє собою матрицю порівняння сильних, слабких

сторін, загроз та можливостей на основі раніше виділених сильних та слабких сторін, ринкових загроз та можливостей.

Перелік ринкови загроз та можливостей складаємо на основі визначених раніше (Таблиця 4.6).

Варто зауважити що як ринкові можливості, так і ринкові загрози є прогнозованими результатами впливу ще не реалізованих на ринку факторів, та мають імовірність здійснення.

Таблиця 4.11 – SWOT-аналіз пропонованого стартап-проекту

Сильні сторони: Динамічна робота Робота у контексті двомовного середовища	Слабкі сторони: Низька швидкодія
Можливості: Нецільова популярність Відсутність активної конкуренції	Загрози: Низький попит Боротьба ЗМІ з аналізом їх текстів

Таблиця 4.13 – Альтернативи ринкововго впровадження стартап-проекту

	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Орієнтація на розробку браузерного додатку застереження користувачів	60%	500 люд.-год.

Продовження таблиці 4.13

2	Орієнтація на розробку фільтраційно-модераційної системи	60%	300 люд.-год.

4.4 Розроблення ринкової стратегії проекту

У даному підрозділі проаналізовано ринкові стратегії, визначено стратегії охоплення ринку, описано цільові групи потенційних споживачів (таблиця 4.13).

Таблиця 4.14 – Цільові групи потенційних споживачів

	Опис цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Технічно обізнані люди, які бажають захистити родичів від небезпечних новин (ринок персонального захисту)	Готові	Високий	Низька	Просто

Продовження таблиці 4.14

2	Інформаційні бюро, агенства, тощо, які хочуть пришвидшити та автоматизувати частину своєї роботи.	Потребують переговорів	Високий	Середня	Складно
3	Ринок автоматичних модераційних систем	Потребують переговорів	Високий	Середня	Середньо

За результатами аналізу цільових груп потенційних споживачів, яким пропонується дана система було обрано ринок персонального захисту від дезінформації та ринок автоматичних модераційних систем, та сформовано базову стратегію розвитку (таблиця 4.14).

Таблиця 4.15 – Базова стратегія розвитку

	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Орієнтація на ринок персонального захисту	Стратегія концентрованого маркетингу	Технічно обізнані люди, які бажають захистити родичів від небезпечних новин	Концентрація

Продовження таблиці 4.15

2	Орієнтація на ринок автоматичних модераційних систем	Стратегія концентрованого маркетингу	Чат-сервіси, форуми, інші ресурси, що наповнюються користувачами.	Вертикальна інтеграція

Наступним кроком виберемо стратегію конкурентної поведінки на ринку (таблиця 4.16).

	Чи є пропонований проект «першопрохідцем» на ринку?	Чи буде дана компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде дана компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Шукати нових споживачів	Працювати над оптимізацією роботи за допомогою впровадження централізованої бази даних та додавання аналітичних метрик, які дозволять прийняти рішення раніше	Стратегія заняття конкурентної ніші

Остаточним кроком розробки ринкової стратегії пропонованого проекту є стратегія позиціонування, яка полягає у формуванні ринкової позиції, яка є для проекту є ідентифікуючою серед користувачів. Стратегія позиціонування зазначена у таблиці 4.17.

Таблиця 4.17 – Стратегія позиціонування

	Вимоги до товару від цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Динаміччність та гнучкість роботи. Легка інтеграція (для користувачів інтегрованого сервісу). Легкість використання.	Концентрація	Технологічно та інформаційно освідчені люди потребують доступного способу захисту близьких людей від неякісної інформації.	Легкість використання.

4.5 Розроблення маркетингової програми проекту

У таблиці 4.18 наведено попередній аналіз конкуретноспроможності пропонованого продукту на ринку.

Таблиця 4.18 – Ключові переваги концепції потенційного товару

	Потреба	Вигода	Перевага (існуюча або майбутня)
1	Динамічна робота (з будь-якими ресурсами та новинами	Надійність продукту (працюватиме завжди і з усім що дають)	Основний конкурент обмежений у кількості підтримуваних ресурсів
2	Робота з українським двомовним контекстом	Продукт працює з будь-яким українським новинним ресурсом	Конкурент TrustedTimes підтримує тільки англійську

Далі розроблена трирівнева маркетингова модель продукту, наведена у таблиці 4.18.

Таблиця 4.18 – Трирівнева модель продукту

Рівні продукту			
I. Товар за задумом	Витягнення та класифікація новинного матеріалу в браузері в режимі реального ачсу		
II. Товар у реальному виконанні	Властивість/характеристика	М/Нм	Вр/Тх/Тл/Ор
	Можливість оптимізації витрат часу	М	Тл
	Можливість оптимізації ергономіки	Нм	Е

Продовження таблиці 4.18

II. Товар у реальному виконанні	Властивість/характеристика	М/Нм	Вр/Тх/Тл/Ор
II. Товар у реальному виконанні	Відповідність актуальним технологіям	М	Тх
	Відповідає вимогам ДСТУ ISO/IEC 25030:2015 Програмна інженерія. Вимоги щодо якості та оцінювання програмного продукту (SQuaRE).		
	Пакування: готовий до використання інсталятор, доступність до прямого встановлення через Chrome Web Store		
II. Товар у реальному виконанні	Марка: Fakebasher		
III. Товар із підкріпленням	Потенційний користувач може ознайомитись з роботою системи використовуючи випробувальний період у 30 днів.		
Захист від копіювання	Підписаний інсталятор. Захист пропрієнтарними ліцензіями. Захист іншими методами захисту інтелектуальної власності.		

Розшифровка скорочень таблиці 4.18:

- М/Нм – монотонні або немонотонні;
- Вр/Тх/Тл/Е/Ор – вартісні, технічні, технологічні, ергономічні або органолептичні(останній – для продуктів харчування)

Захист інтелектуальної власності продукту є важливим моментом публікування та комерційності діяльності. Такий захист може бути досягнуто за допомогою DRM, захисту ідеї, ноу-хау, патентування, тощо.

Наступним етапом є визначення системи збуту, зазначене у таблиці 4.20.

Таблиця 4.20 – Формування системи збуту

	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Певний початковий період користування продуктом має бути безкоштовним	Легкість у сплаті послуг, легкість у встановленні	Розробник – магазин – користувач	Проведення збуту через посередника необхідне, оскільки додаток є залежним від системи розповсюдження платформи

Наступним та останнім кроком у розробці маркетингової програми є концепція маркетингових комунікацій, спираючись на визначену специфічність планованих клієнтів та основу позиціонування.

Таблиця 4.21 – Концепція маркетингових комунікацій

	Специфіка поведінки цільових клієнтів	Канали комунікацій, що використовують цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Підписуються на програмний продукт	Інтернет	Робота з будь-якими ресурсами англійською та українською	Довести, що додаток захистить близьких від небажаних новин	-

4.6 Висновки до розділу

Пропонований продукт є конкурентоздатним, оскільки має істотні переваги над існуючими конкурентами. Було визначено шляхи для подальшого розвитку, шляхи збуту та описано маркетингову стратегію. Основною цільовою аудидторією було обрано технологічно обізнаних людей, яких непокоїть доступ менш освідчених близьких до неякісних новинних матеріалів.

ВИСНОВКИ

Було проаналізовано існуючі підходи та програмні засоби і сервіси, вже присутні на ринку екстракції структурованої інформації. Було проаналізовано сферу аналізу новин на предмет вмісту маніпулятивної та дезінформативної інформації, окреслено існуючі сервіси та підходи в аналізі.

Було розроблено простий до імплементації алгоритм екстракції структурованої інформації, який суміщує в собі процеси краулінгу та скрапінгу, таким чином спрощуючи інтеграційні та експлуатаційні процеси використання алгоритму, а також окреслено загальний алгоритм підготовки текстових даних до аналізу методами машинного навчання.

Було оглянуто та викладено математичні засади деяких алгоритмів векторизації даних та відповідних методів машинного навчання, які застосовуються для обробки та класифікації текстових даних.

Окреслені алгоритми екстракції підготовки, векторизації та класифікації було розроблено. Результати реалізації вищезазначених алгоритмів порівняно: алгоритм екстракції було порівняно з існуючим на ринку найближчим аналогом за результатами екстракції та за результатами тренування алгоритмів машинного навчання на видобутих даних.

На прикладі розробленого екстрактора було доведено, що прості у розробці та підтримці загальні алгоритми екстракції структурованої інформації є конкурентноспроможними з існуючими високотехнологічними системами за рахунок простоти імплементації, підтримки та цінових характеристик.

На прикладі розроблених алгоритмів машинного навчання було доведено, що існуючі засоби машинного навчання здатні до повноцінної роботи та класифікації текстів на масиві даних, який включає в себе тексти українською та російською мовами, а також що результати тренування класичних алгоритмів машинного навчання є співставними з більш складними у розробці та тренуванні нейронними мережами довгої короткочасної пам'яті.

Етапи роботи було опубліковано у статті «Класифікація новин за достовірністю на основі методів машинного навчання»[19].

ПЕРЕЛІК ПОСИЛАНЬ

1. P. H. Cording (2011). Algorithms for Web Scraping.
2. P YesuRaju and P KiranSree (2013). A language independent web data extraction using vision based page segmentation algorithm.
3. Саханда П.П. (2019). Система виявлення веб-скраперів з використанням пасток
4. Zehuan Cai, Jin Liu, Lamei Xu, Chunyong Yin, Jin Wang (2017). A Vision Recognition Based Method for Web Data Extraction. Advanced Science and Technology Letters Vol.143 (AST 2017), pp.193-198
5. scrapinghub.com/whitepapers/
6. М. Кіца (2017). Особливості та методи виявлення фейкової інформації в українських ЗМІ.
7. Jiawei Zhang and Bowen Dong and Philip S. Yu (2019). FAKEDETECTOR: Effective Fake News Detection with Deep Diffusive Neural Network
8. В.В. Жуковська, (2013). Вступ до корпусної лінгвістики. Вид-во ЖДУ ім. І.Франка.
9. Jayashree M Kudari, Varsha V, Monica BG, Archana R (2020). Fake News Detection using Passive Aggressive and TF-IDF Vectorizer. International Research Journal of Engineering and Technology (IRJET) Volume: 07 Issue: 09 | Sep 2020. e-ISSN: 2395-0056 p-ISSN: 2395-0072.
10. Silva, C.. (2003). The importance of stop word removal on recall values in text categorization. 3. 1661 - 1666 vol.3. 10.1109/IJCNN.2003.1223656.
11. Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer (2006). Online Passive-Aggressive Algorithms. Journal of Machine Learning Research 7 (2006) 551–585
12. S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. Neural Computation, 9(8):1735–1780, 1997.

13. Basaldella, Marco & Antolli, Elisa & Serra, Giuseppe & Tasso, Carlo. (2018). Bidirectional LSTM Recurrent Neural Network for Keyphrase Extraction. 10.1007/978-3-319-73165-0_18.
14. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001)
15. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural Networks 18(5), 602–610 (2005)
16. Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva. 2017. Energy efficiency across programming languages: how do energy, time, and memory relate? In Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering (SLE 2017). Association for Computing Machinery, New York, NY, USA, 256–267.
17. Lei, Kai & Ma, Yining & Tan, Zhi. (2014). Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js. 661-668. 10.1109/CSE.2014.142.
18. Дмитро Чоповський, Інститут масової інформації., 2011. Журналістські стандарти: інформаційна довідка. Доступно за посиланням https://imi.org.ua/monitorings/jurnalistski-standarti-informatsiyna-dovidka-i28623
19. Смілянець Ф.А., Мажара О.О (2020). Класифікація новин за достовірністю на основі методів машинного навчання. IV Всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020).

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

scraper.js

```
#!/usr/bin/env node

const puppeteer = require('puppeteer');
const fs = require('fs');
const commandLineUsage = require('command-line-usage')
const commandLineArgs = require('command-line-args');
const robotsParser = require('robots-txt-parser');

const sections = [
  {
    header: 'NewsScraper',
    content: 'A greedy scraper-crawler for news websites'
  },
  {
    header: 'Options',
    optionList: [
      {
        name: 'domain',
        typeLabel: '{underline link to a website root}',
        description: 'The website to crawl-scrape.',
        defaultOption: true,
        alias: 'd',
      },
      {
        name: 'exclude',
        typeLabel: '{underline string}',
        description: 'The strings that if found in a link will ignore the link',
        alias: 'e',
        multiple: true,
      },
      {
        name: 'output',
        typeLabel: '{underline file}',
        description: 'The file to write the results to. Defaults to
scraped.json',
        alias: 'o',
      },
      {
        name: 'sleep',
        typeLabel: '{underline ms}',
        description: 'How much to wait between requests',
        alias: 's',
      },
      {
        name: 'useragent',
        description: 'Useragent to check against when looking up robots.txt.
Defaults to Googlebot',
        alias: 'u',
      },
      {
        name: 'help',
        description: 'Print this usage guide.'
      }
    ]
  }
]
```

```
const optionDefinitions = [
  { name: 'domain', alias: 'd', type: String, defaultOption: true },
  { name: 'exclude', alias: 'e', type: String, multiple: true, defaultValue: []
},
  { name: 'output', alias: 'o', type: String, defaultValue: 'scraped.json' },
  { name: 'useragent', type: String, defaultValue: 'Googlebot' },
  { name: 'sleep', alias: 's', type: Number, defaultValue: 500 },
  { name: 'help', type: Boolean }
];

const options = commandLineArgs(optionDefinitions);
const usage = commandLineUsage(sections);

if (options.help) {
  console.log(usage);
  process.exit();
}
if (!options.domain) {
  console.error('Please provide an url to crawl');
  process.exit();
}

// process.exit();

const MAXIMUM_STRIPPED_DOMAIN_PLACEMENT_IN_LINK = 'https://www.'.length + 1; //
for it to be from the domain we need
const HEADER_THRESHOLD = 20;
const CONTENT_THRESHOLD = 1000;

const visited = new Map();
const links = new Set([options.domain]);

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  const robots = robotsParser({
    userAgent: options.useragent,
    allowOnNeutral: false
  });

  await robots.useRobotsFor(options.domain);
  const iterator = links.values();

  for (const currentUrl of iterator) {
    await sleep(options.sleep);
    links.delete(currentUrl);
    try {
      if (visited.has(currentUrl)) {
        continue;
      }
      console.log(`CURRENT PAGE: ${currentUrl}, LINKS TO GO: ${links.size}`);

      await page.goto(currentUrl);

      const lang = await page.$eval('html', element => element.lang);

      if (lang !== 'uk' && lang !== 'ru' && lang !== 'ru-RU' && lang !== '' &&
lang !== undefined) {
        visited.set(currentUrl, { isArticle: false, reason: 'Wrong lang' });
        continue;
```

```
      }

      const strippedUrl = stripUrl(currentUrl);
      const newLinks = await page.$$eval('a', (elements, strippedUrl) =>
elements.map(element => {
        const link = element.href.split('#')[0];
        if (link.startsWith('/')) {
          return `${strippedUrl}${link}`
        } else {
          return link;
        }
      }), strippedUrl);

      let header = await page.$$eval('h1, h2.title, h2.post_name', (elements,
HEADER_THRESHOLD) => {
        return elements.map(element => element.textContent).filter(el =>
el.length > HEADER_THRESHOLD).reduce((acc, el) => el.length > acc.length ? el :
acc, '');
      }, HEADER_THRESHOLD);

      if (!header) {
        header = await page.$$eval('h2', (elements, HEADER_THRESHOLD) => {
          return elements.map(element => element.textContent).filter(el =>
el.length > HEADER_THRESHOLD).reduce((acc, el) => el.length > acc.length ? el :
acc, '');
        }, HEADER_THRESHOLD);
      }
      console.log(header);

      if (header && header.length > HEADER_THRESHOLD) {
        const articleContent = await page.$$eval(
          'p, article div',
          elements => elements.map(element => {
            element.querySelectorAll('a').forEach(el => el.remove());
            element.querySelectorAll('button').forEach(el => el.remove());
            element.querySelectorAll('script').forEach(el => el.remove());
            element.querySelectorAll('style').forEach(el => el.remove());
            return element.textContent.split('\n').filter(el => el).join('
').split('\t').filter(el => el).join(' ');
          }).join(' ')
        );
        console.log(`HEADER ${header} CONTENT ${articleContent.length}`);
        if (articleContent && articleContent.length > CONTENT_THRESHOLD) {
          visited.set(currentUrl, {
            isArticle: true,
            header,
            articleContent
          });
        } else {
          visited.set(currentUrl, { isArticle: false, reason: 'Not enough words'
});
        }
      } else {
        visited.set(currentUrl, { isArticle: false, reason: 'No header' });
      }

      newLinks
        .filter(link => {
          const strippedIndexof = link.indexOf(strippedUrl);
          return strippedIndexof !== -1 &&
            strippedIndexof < MAXIMUM_STRIPPED_DOMAIN_PLACEMENT_IN_LINK &&
```

```
            !links.has(link) &&
            !visited.has(link) &&
            !options.exclude.reduce((acc, val) => acc || link.includes(val),
false) &&
            robots.canCrawlSync(link) &&
            link.startsWith('http');
        })
        .map(link => links.add(link));

      fs.writeFile(options.output,
JSON.stringify(Array.from(visited.entries()).filter(entry =>
entry[1].isArticle)), () => {});
    } catch (e) {
      console.log(e)
    }
  }

  await browser.close();
})();



function stripUrl(url) {
  return url.split('http://').join('').split('https://').join('').split('/')[0];
}

function sleep(ms) {
  return new Promise((resolve) => {
    setTimeout(resolve, ms);
  });
}

background.js
import 'babel-polyfill';
import * as tf from '@tensorflow/tfjs';

class Tokenizer {
  constructor(config = {}) {
    this.filters = config.filters || /[\\.,/#!$%^&*;:{}=\-_`~()]/g;
    this.lower = typeof config.lower === 'undefined' ? true : config.lower;

    // Primary indexing methods. Word to index and index to word.
    this.wordIndex = {};
    this.indexWord = {};

    // Keeping track of word counts
    this.wordCounts = {};
  }

  cleanText(text) {
    if (this.lower) text = text.toLowerCase();
    return text
      .replace(this.filters, '')
      .replace(/\s{2,}/g, ' ')
      .split(' ');
  }

  fitOnTexts(texts) {
    texts.forEach(text => {
      text = this.cleanText(text);
      text.forEach(word => {
```

```
        this.wordCounts[word] = (this.wordCounts[word] || 0) + 1;
      });
    });

    Object.entries(this.wordCounts)
      .sort((a, b) => b[1] - a[1])
      .forEach(([word, number], i) => {
        this.wordIndex[word] = i + 1;
        this.indexWord[i + 1] = word;
      });
  }

  textsToSequences(texts) {
    return texts.map(text => this.cleanText(text).map(word =>
this.wordIndex[word] || 0));
  }

  textToSequence(text) {
    const a = text
    .replace(/([^\p{L}])/giu, ' ')
    .replace(/\s+/g, ' ')
    .toLowerCase()
    .split(' ')
    .filter(word => !this.stopwords.has(word) && this.wordIndex[word])
    console.log(a)
    return a.map(word => this.wordIndex[word])
    .filter(word => word < this.numWords);
  }

  toJson() {
    return JSON.stringify({
      wordIndex: this.wordIndex,
      indexWord: this.indexWord,
      wordCounts: this.wordCounts
    })
  }
}

const tokenizerFromJson = (js, stopwords) => {
  console.log(js)
  const tokenizer = new Tokenizer();
  tokenizer.wordIndex = JSON.parse(js.config.word_index);
  tokenizer.indexWord = JSON.parse(js.config.index_word);
  tokenizer.wordCounts = JSON.parse(js.config.word_counts);
  tokenizer.numWords = js.config.num_words;
  tokenizer.stopwords = new Set(stopwords);
  return tokenizer;
};

function padSequence(sequence, maxLength) {
  if (sequence.length > maxLength) {
    return sequence.slice(0, maxLength);
  } else {
    return [
      ...(new Array(maxLength - sequence.length)).fill(0),
      ...sequence
    ];
  }
}

/**
```

```
 * Async loads a mobilenet on construction.  Subsequently handles
 * requests to classify images through the .analyzeImage API.
 * Successful requests will post a chrome message with
 * 'IMAGE_CLICK_PROCESSED' action, which the content.js can
 * hear and use to manipulate the DOM.
 */
class FakeClassifier {
  constructor() {
    this.loadModel();
  }

  /**
   * Loads mobilenet from URL and keeps a reference to it in the object.
   */
  async loadModel() {
    try {
      this.model = await tf.loadLayersModel('weights/model.json');
      const tokenizerData = await (await
fetch(chrome.runtime.getURL('weights/tokenizer.json'))).json();
      const stopwords = await (await
fetch(chrome.runtime.getURL('stopwords_ua.json'))).json();
      console.log(stopwords)
      this.token = tokenizerFromJson(tokenizerData, stopwords);
      console.log(this.token)
      console.log(this.model);
    } catch (e) {
      console.error(e);
    }
  }

  async predict(articleText) {
    console.log('Predicting...');
    const startTime1 = performance.now();
    let startTime2;

    const output = tf.tidy(() => {
      const sequence = this.token.textToSequence(articleText);
      console.log(sequence);
      startTime2 = performance.now();

      console.log(this.model.input.shape[1], padSequence(sequence,
this.model.input.shape[1]));
      const inputTensor =
      tf.tensor(
        [padSequence(sequence, this.model.input.shape[1])],
        [1, this.model.input.shape[1]]
      );
      console.log(inputTensor.arraySync);
      return this.model.predict(inputTensor
      );
    });

    // Convert logits to probabilities and class names.
    const totalTime1 = performance.now() - startTime1;
    const totalTime2 = performance.now() - startTime2;
    console.log(
        `Done in ${totalTime1.toFixed(1)} ms ` +
        `(not including preprocessing: ${Math.floor(totalTime2)} ms)`);
    return output.array();
  }
}
```

```
const fakeClassifier = new FakeClassifier();
console.log(padSequence);
console.log(tf.tensor);
console.log(fakeClassifier)

chrome.runtime.onMessage.addListener(onMessage.bind(this));

function onMessage(message, sender, sendResponse) {
  console.log('asdasdasdasdas', sender, message)
  if (message.action === 'TF_ESTIMATE_ARTICLE') {
    fakeClassifier.predict(message.article.article).then(result => {
      console.log(result);

      sendResponse(result[0][0]);
    });

    return true;
  }
}

if (!window.afContentScriptLoaded && window.location === window.parent.location)
{
  const HEADER_THRESHOLD = 20;
  const CONTENT_THRESHOLD = 1000;

  chrome.runtime.onMessage.addListener((message, sender, sendResponse) => {
    if (message && message.action === 'ESTIMATE_ARTICLE') {
      const article = findArticle();
      if (article) {
        chrome.runtime.sendMessage({
          action: 'TF_ESTIMATE_ARTICLE',
          article: findArticle()
        }, (response) => {
          sendResponse(response);
          console.log(response);
        });
      } else {
        sendResponse(null);
      }
    }
  });

  function findArticle() {
    const lang = document.getElementsByTagName('html')[0].lang;
    if (lang && lang !== 'ua' && lang !== 'uk' && lang !== 'ru' && lang !==
'ru_RU') {
      return null;
    }

    const header = getArticleHeader();

    if (!header || header.length < HEADER_THRESHOLD) {
      return null;
    }

    const article = getArticleContent();

    if (!article || article.length < CONTENT_THRESHOLD) {
      return null;
    }
```

```
    return { header, article };
  }

  window.afContentScriptLoaded = true;

  document.addEventListener('DOMContentLoaded', () =>
setTimeout(onPageLoad.bind(this), 300));

  function getArticleHeader() {
    let header = Array
      .from(document.getElementsByTagName('h1'))
      .map(element => element.textContent)
      .filter(el => el.length > HEADER_THRESHOLD)
      .reduce((acc, el) => el.length > acc.length ? el : acc, '');

    if (!header) {
      header = Array
        .from(document.getElementsByTagName('h2'))
        .map(element => element.textContent)
        .filter(el => el.length > HEADER_THRESHOLD)
        .reduce((acc, el) => el.length > acc.length ? el : acc, '');
    }

    return header;
  }

  function getArticleContent() {
    let article = Array.from(document.getElementsByTagName('p'))
      .map(node => node.cloneNode(true))
      .map(element => {
        element.querySelectorAll('a').forEach(el => el.remove());
        element.querySelectorAll('button').forEach(el => el.remove());
        element.querySelectorAll('script').forEach(el => el.remove());
        element.querySelectorAll('style').forEach(el => el.remove());
        element.querySelectorAll('img').forEach(el => el.remove());

        return element.textContent.replace(/\n|\t/g, ' ')
      }).join(' ');

    if (!article || article.length < CONTENT_THRESHOLD) {
      article = Array.from(document.getElementsByTagName('article'))
        .map(node => node.cloneNode(true))
        .map(element => {
          element.querySelectorAll('a').forEach(el => el.remove());
          element.querySelectorAll('button').forEach(el => el.remove());
          element.querySelectorAll('script').forEach(el => el.remove());
          element.querySelectorAll('img').forEach(el => el.remove());
          element.querySelectorAll('style').forEach(el => el.remove());

          return element.textContent.replace(/\n|\t/g, ' ')
        }).join(' ');
    }
  }

  function onPageLoad() {
    const article = findArticle();
    if (article) {
      console.log(article)
      chrome.runtime.sendMessage({
        action: 'TF_ESTIMATE_ARTICLE',
```

```
        article
      }, (response) => onAnalysisReady(response));
    }
  }

  function onAnalysisReady(response) {
    console.log(response);
    if (!response) {
      return;
    }

    if (response < 0.5) {
      const newDiv = document.createElement('div');
      newDiv.style.width = '100%';
      newDiv.style.lineHeight = '32px';
      newDiv.style.fontSize = '24px';
      newDiv.style.padding = '16px';
      newDiv.style.background = 'red';
      newDiv.style.color = 'white';
      newDiv.id = 'fakeBasherLoadingTag';
      newDiv.style.position = 'fixed';
      newDiv.style.zIndex = 100000;
      newDiv.innerText = 'Увага! Ця стаття дуже схожа на статті "сміттєвих" ЗМІ.
Скоріш за все вона є маніпулятивною/фейковою та намагається сформувати у вас
потрібну авторові думку щодо чогось або когось. Не радимо читати цей матеріал.';

      document.body.prepend(newDiv);
    } else if (response < 0.7) {
      const newDiv = document.createElement('div');
      newDiv.style.width = '100%';
      newDiv.style.lineHeight = '32px';
      newDiv.style.fontSize = '24px';
      newDiv.style.padding = '16px';
      newDiv.style.background = 'orange';
      newDiv.style.position = 'fixed';
      newDiv.style.zIndex = 100000;
      newDiv.id = 'fakeBasherLoadingTag';
      newDiv.innerText = 'Обережно! Ця стаття не дуже схожа на статті якісних
ЗМІ. Можливо, вона є маніпулятивною/фейковою. Рекомендуємо ретельно перевірити
викладену інформацію, або звернутись до іншого ЗМІ.';

      document.body.prepend(newDiv);
    } else {
      const newDiv = document.createElement('div');
      newDiv.style.width = '100%';
      newDiv.style.lineHeight = '32px';
      newDiv.style.fontSize = '24px';
      newDiv.style.padding = '16px';
      newDiv.style.background = 'yellow';
      newDiv.style.position = 'fixed';
      newDiv.id = 'fakeBasherLoadingTag';
      newDiv.style.zIndex = 100000;
      newDiv.innerText = 'Текст схожий на тексти якісних ЗМІ. Алгоритм може бути
неправий, тому завжди перевіряйте будь-яку інформацію!';

      document.body.prepend(newDiv);
    }
  }
}
```

UNICHECK

Ім'я користувача:
Попенко Володимир Дмитрович

ID перевірки:
1005447537

Дата перевірки:
14.12.2020 00:12:19 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
14.12.2020 01:57:36 EET

ID користувача:
77149

Назва документа: **Smiljanets_magistr_ip92mp**

Кількість сторінок: **104** Кількість слів: **15495** Кількість символів: **125031** Розмір файлу: **3.96 MB** ID файлу: **1005737894**

5.56%
Схожість

Найбільша схожість: **1.96%** з джерелом з Бібліотеки (ID файлу: **1000765142**)

4.01% Джерела з Інтернету	185	Сторінка 106
3.73% Джерела з Бібліотеки	460	Сторінка 109

1.06% Цитат

Цитати	10	Сторінка 110

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	10